SZIKSZAI CSABA

hobbi

Barátunk, a számítógép



Az ügyeskedni, „alkotni” vágyó fiataloknak szól a Móra Kiadó új, foglalkoztató sorozata, a Hobbi. A lányok és a fiúk érdeklődési körére gondolva állítottuk össze témáit, így szerepel majd köztük kötés, horgolás, vasút- és hajómodellezés, játékkészítés és fényképészeti ábécé, számítógép-programozás és textilmunka.

Szép és hasznos hobbi, ha állatot tartunk a lakásban. A kezdő állatbarátok számára is összeállítottunk egy kötetet, amelyben kis kedvenceik gondozására, tartására kapnak sok hasznos tanácsot.

Minden Hobbi könyv gazdag illusztrációs anyaggal – rajzzal, fényképpel – jelenik meg.

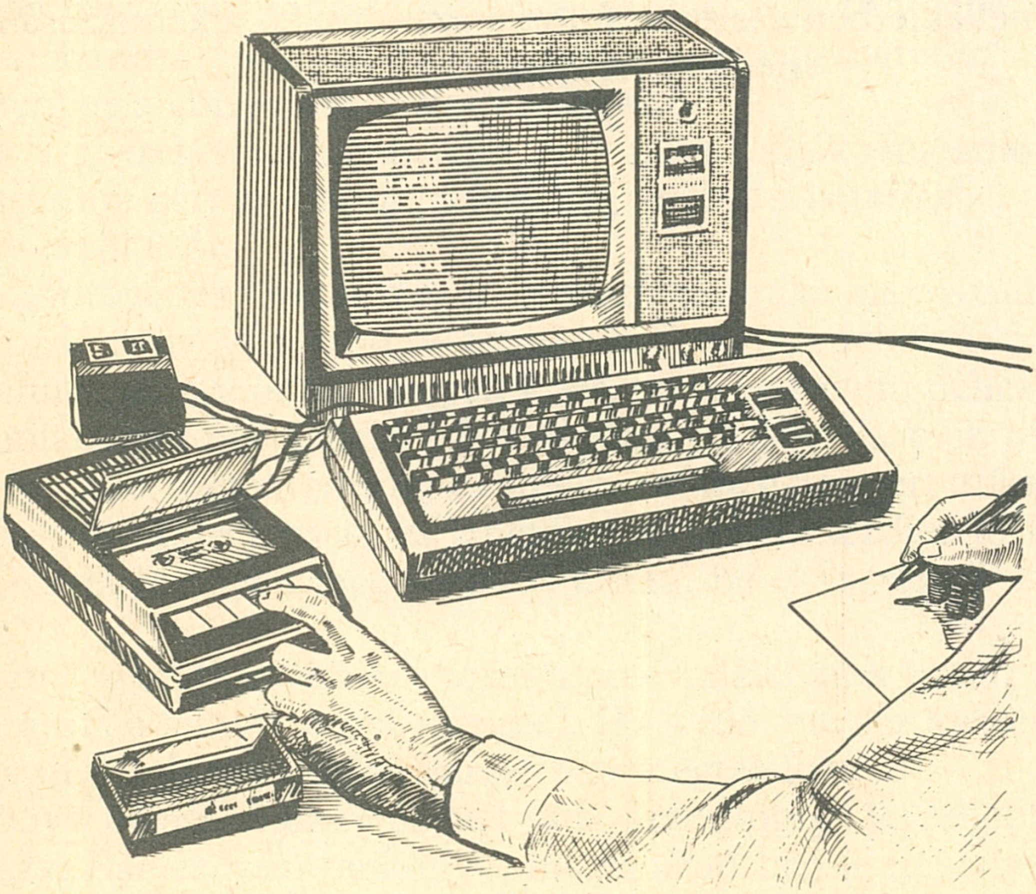
SZIKSZAI CSABA

Barátunk, a számítógép

SOROZATSZERKESZTŐ
KARÁDI ILONA

SZIKSZAI CSABA

Barátunk, a számítógép



MÓRA FERENC KÖNYVKIADÓ

A FEDÉL VIDA GYŐZŐ MUNKÁJA
HEGYI GÁBOR ÉS KÁDÁR KATA DIÁINAK FELHASZNÁLÁSÁVAL

MOLNÁR OTTÓ RAJZAIVAL

A fedél felvételeinek elkészítéséhez
az IBM Fejlesztő Számítóközpontja nyújtott segítséget

A mellékletben szereplő fotók a számítógép alkalmazásának
néhány területét mutatják be

ELŐSZÓ

Kedves Gyerekek!

Milyen lehet és hogyan működik a számítógép? Biztosan feltettétek már ezt a kérdést, amikor meghallottátok, hogy egy ilyen gép egyetlen másodperc alatt több százezer összeadást is el tud végezni.

Aki már tudja a választ, az nyugodtan abbahagyhatja itt az olvasást, mert ez a könyv akkor nem neki való!

Azok a gyerekek azonban, akik még nem tudják, fogjanak hozzá az olvasáshoz bátran – amint az én gyermekeim is tették –, mert ez a könyv erről szól.

A számítógépek építésével a számítástechnika nevű tudomány foglalkozik. Művelői között matematikusokat, mérnököket, fizikusokat találunk, de ebből ne higgyétek azt, hogy ez valami nehéz dolog. A tudományban nincsenek nehéz és könnyű dolgok, csak olyanok, amelyeket már megértettünk, és amelyeket még nem értettünk meg.

Olvassátok figyelmesen, és lehetőleg az elején kezdjétek. Bármelyik fejezetet könnyen meg fogjátok érteni, ha elolvastátok az előtte lévőt is.

Néhány számítástechnikai fogalommal találkoztok majd, melyek magyarázata a könyv végén szerepel. Ha a szó magyar kiejtése más, akkor a helyes kiejtés utána zárójelben megtalálható.

A könyv olvasásához nem kell saját számítógép, legfeljebb papír és ceruza. Bár ez nem tankönyv, mégis akad benne néhány feladat az érdeklődő gyerekek számára. A feladatok megoldásait egyébként megtalálhatjátok a könyv végén.

Mit fogtok tudni, ha elolvastátok ezt a könyvet? Nos, megismeritek a korszerű számítógép működési elvét, főbb részeit, a programozás alapjait, és tudtok majd írni saját kis programokat is BASIC nyelven. Ha hozzájuttok egy igazi számítógéphez, akkor nagyon könnyen megtanuljátok majd a kezelését, mert az alapokat már ismeritek. Jobban fog menni a programozás, és biztonsággal eligazodtok az egyre bonyolultabbnak tűnő BASIC utasítások között.

Annak örülnék azonban a legjobban, ha akadnának köztetek olya-

nok, akik kedvet kapnak a számítástechnikához, és akár az iskolájukban, akár a legközelebbi számítástechnikai szakkörben elkezdenék az önálló munkát.

Lehet, hogy köztük lesznek azok is, akik majd megépítik a még korszerűbb számítógépeket?

Lehet. Talán éppen Te leszel az! Kezdjétek hát az olvasást, jó szórakozást, jó ötleteket kíván

Szikszai Csaba

Egy meglepő bejelentés

Az a leghelyesebb, ha az elején kezdem a dolgokat, és bejelentem: a számítógép a kivonást, a szorzást és az osztást is összeadásokkal csinálja!

Legegyszerűbb ezt belátni a kivonás esetében.

Vonjunk ki például nyolcból kettőt!

Az iskolában ezt így tanultuk:

$$\begin{array}{r} 8 \\ -2 \\ \hline 6 \end{array}$$

Ahogy most fogom csinálni, azt nem tanultátok, de így is ugyanaz az eredmény, figyeljetek!

Tízes számrendszerben számolunk. Bármilyen számhoz tudok találni egy másikat, amely 10 valamilyen hatványára (tehát 10-re, 10 · 10-re, 10 · 10 · 10-re stb.) egészíti ki. Az ilyen kiegészítő számot tízes *komplemens*nek hívják.

Például 2 komplemensei a 8, 98, 998 stb. számok, attól függően, hogy 10 melyik hatványára akarjuk a kettest kiegészíteni:

10 hatványai	komplemensek	Így egészíti ki:
10	8	2 + 8 = 10
100	98	2 + 98 = 100
1000	998	2 + 998 = 1000
⋮ stb.	⋮ stb.	⋮ stb.

– Válasszatok most a hatványok közül egyet!

– Bármelyiket lehet választani?

– Természetesen.
– Legyen akkor a százas.

– Jól van. Figyeljetek, most következik a ravasz megoldás:

Ez a feladat

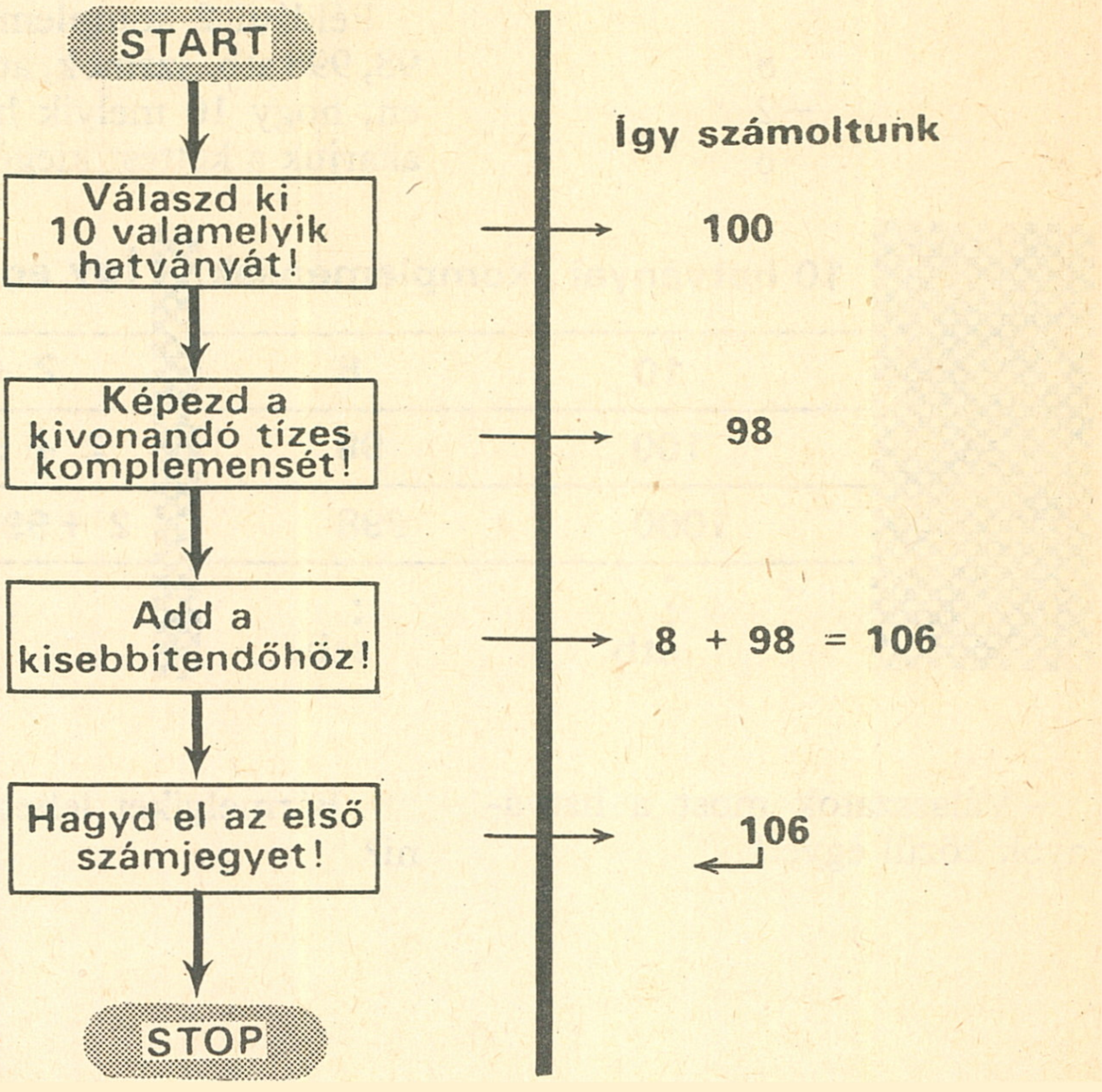
  8 (kisebbítendő)
−2 (kivonandó)
  ?

Képezem a kivonandó tízes komplemensét: ez 98, mert 2+98 = 100.

Hozzáadom a kisebbítendőhöz: 8+98 = 106.

Az összeg (a 106) annyi számjegyből áll, mint a választott hatványunk – a 100 –, ezért az első számjegyet elhagyom: 106 marad a 06, vagyis az eredmény 6.

– Magyarázd el még egyszer, apu! – kérlelt Balázs, ezért felrajzoltam a megoldás *folyamatábráját*:



– Hogyan kell ezt olvasni?

– A starttól kiindulva, a nyilak irányába haladva, szinte kézen fogva vezetnek a vonalak végig a célig.

– Mint Thészeuszt Ariadné fonala – örvendezett Balázs.

– Most 16-ot vonj ki 8-ból! – ravaszkodott Csaba.

– Ez már nehezebb eset lesz, de azért megmutatom. Mivel nem választottál másik hatványt, maradjunk az előbb használt 100-nál.

Ez a feladat

8 (kisebbítendő)
−16 (kivonandó)
?

Képezem a kivonandó tízes komplemensét: ez 84, mert 16+84 = 100.

Hozzáadom a kisebbítendőhöz: 8+84 = 92.

Látjátok, most az összeg kevesebb számjegyből áll, mint a hatványunk, a 100-as. Ez azt jelenti, hogy az eredmény negatív szám lesz.

– Hogyan kapjuk meg ilyenkor az eredményt?

– Az összegnek képezni kell a tízes komplemensét.

Az összeg 92, tízes komplemense pedig 8.

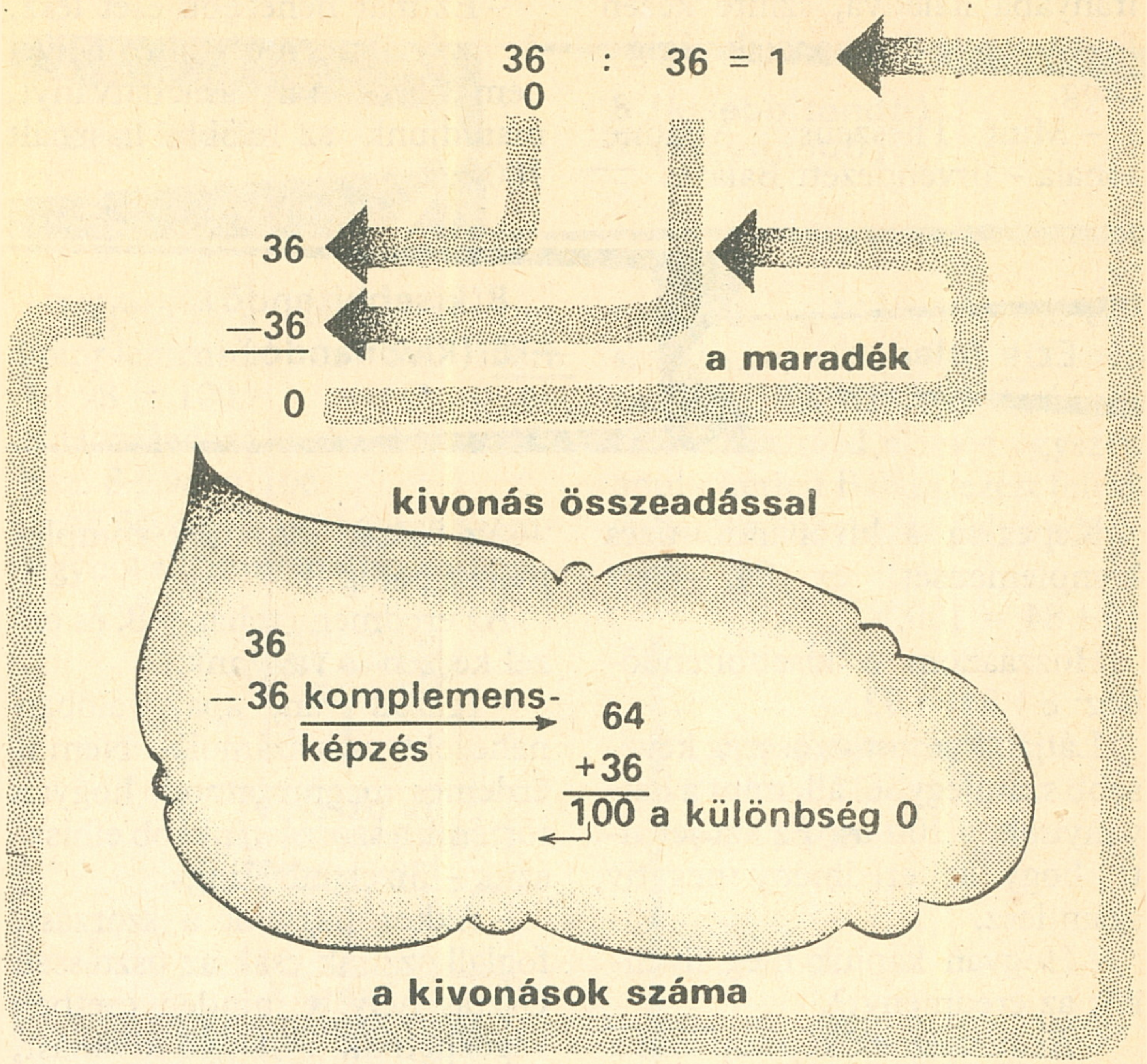
Az eredmény tehát −8, és ezzel készen is vagyunk.

– Igazad van, apa, valóban nehezebb ez a számolás. Nem is érdemes megmutatnod, hogyan történik a szorzás, inkább elhisszük – hízelgett Kinga.

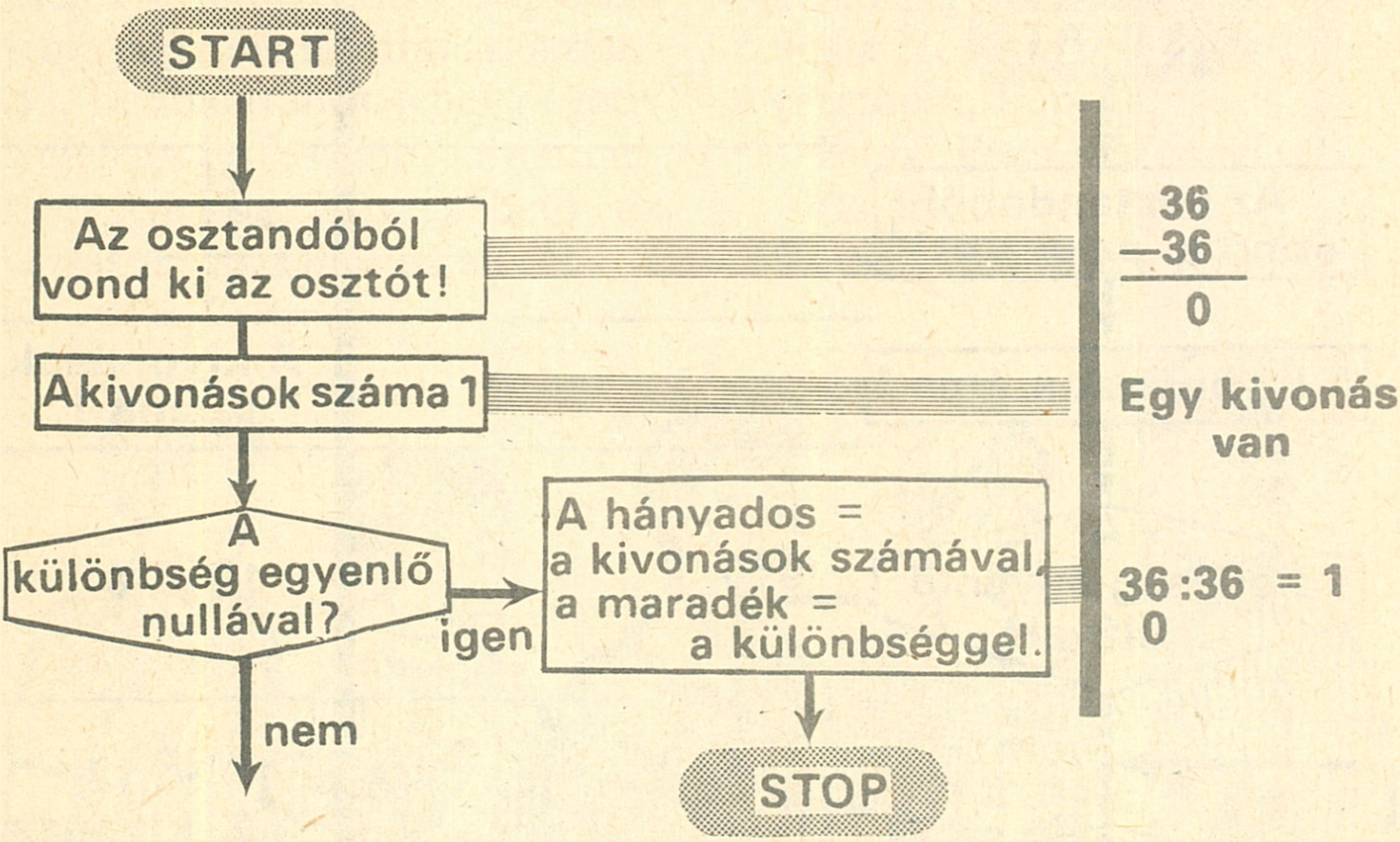
– Akkor hagyjuk a szorzást, foglalkozzunk csak az osztással. Osztás helyett minden esetben végezhetünk ismételt kivonást, egészen addig, amíg a különb-

ség kisebb nem lesz, mint az osztó.

Például a 36 : 36 = 1 osztás így végezhető el kivonással:

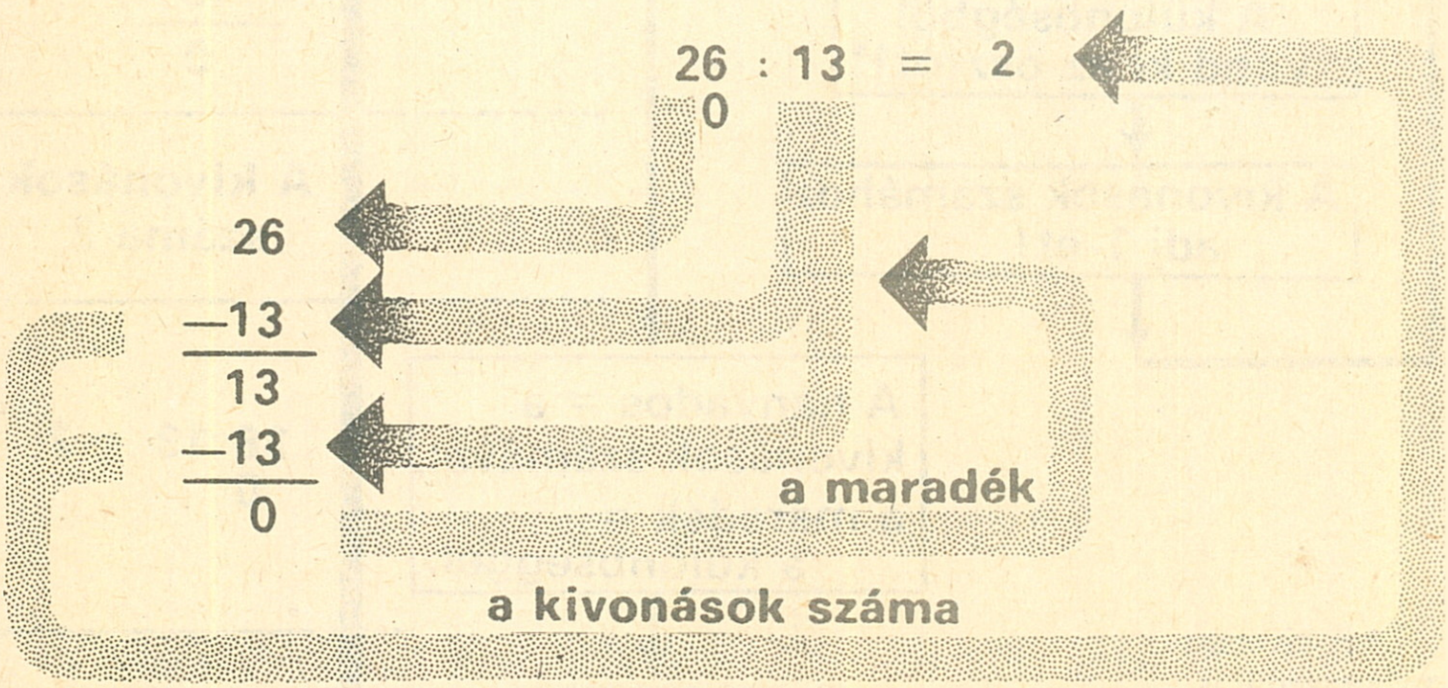


Folyamatábrarészben összefoglalva:

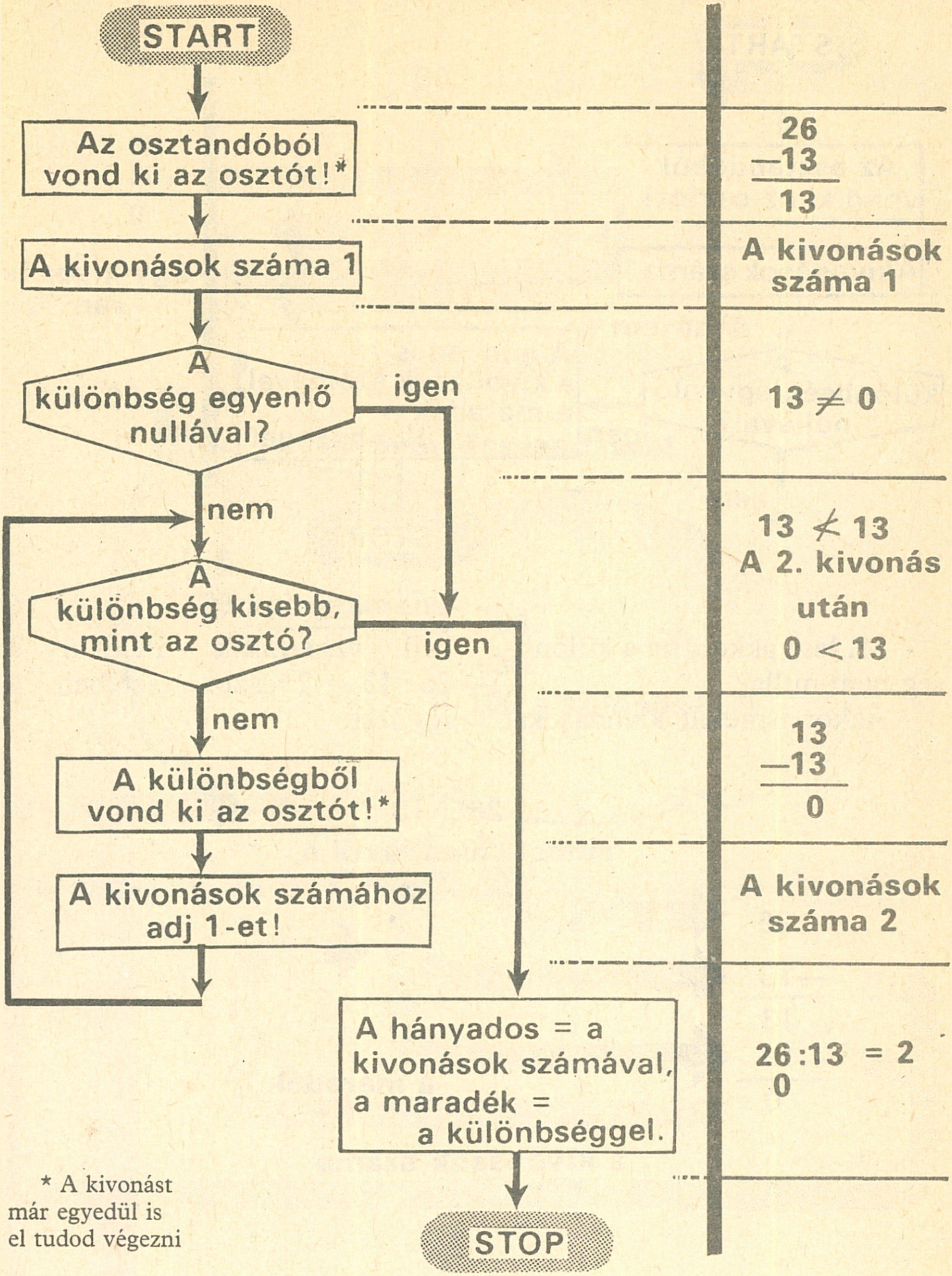


– Mi lesz akkor, ha a különbség nem nulla?

– Akkor ismételt kivonásokat kell végeznünk. Például a 26 : 13 = 2 osztás esetében ez így fest:



Így már be tudjuk fejezni az előbbi folyamatábrát is:



* A kivonást már egyedül is el tudod végezni

– Végre! – sóhajtott Kinga. – Már azt hittem, sose lesz vége!

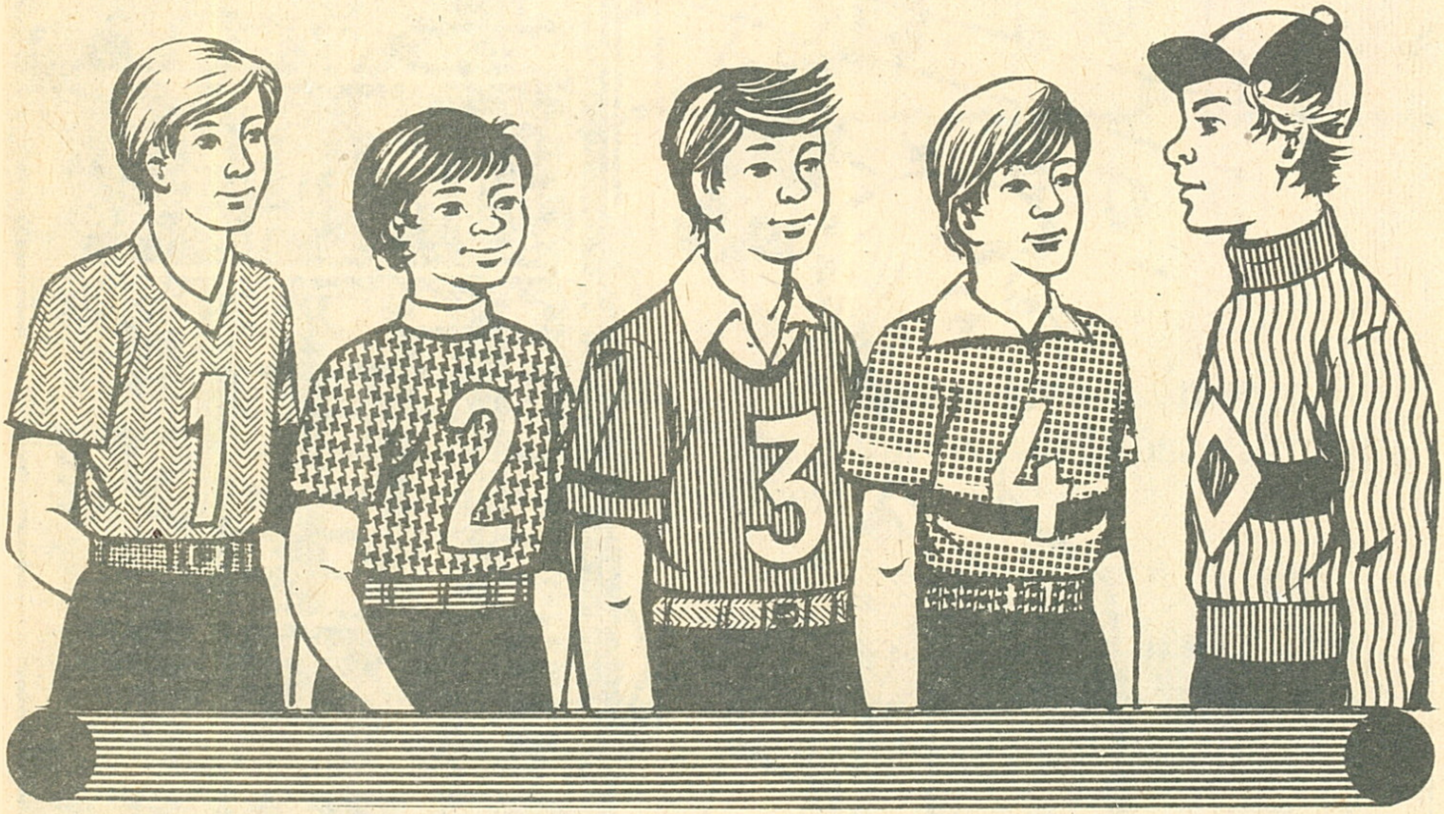
– Ennél jóval bonyolultabb esetek is vannak, hiszen az osztó lehet nagyobb is, mint az osztandó, mindkettőnek lehet negatív előjele és így tovább. Láthattad, hogy valóban elvégezhető az osztás is összeadásokkal. *Nem halogatjuk tovább a dolgot!*

Képzeljétek el, hogy van öt ember, akik együtt dolgoznak. Pista I, Pista II, Pista III, Pista IV és a vezetőjük, Főpista.

– Ennyi Pista már megárt!

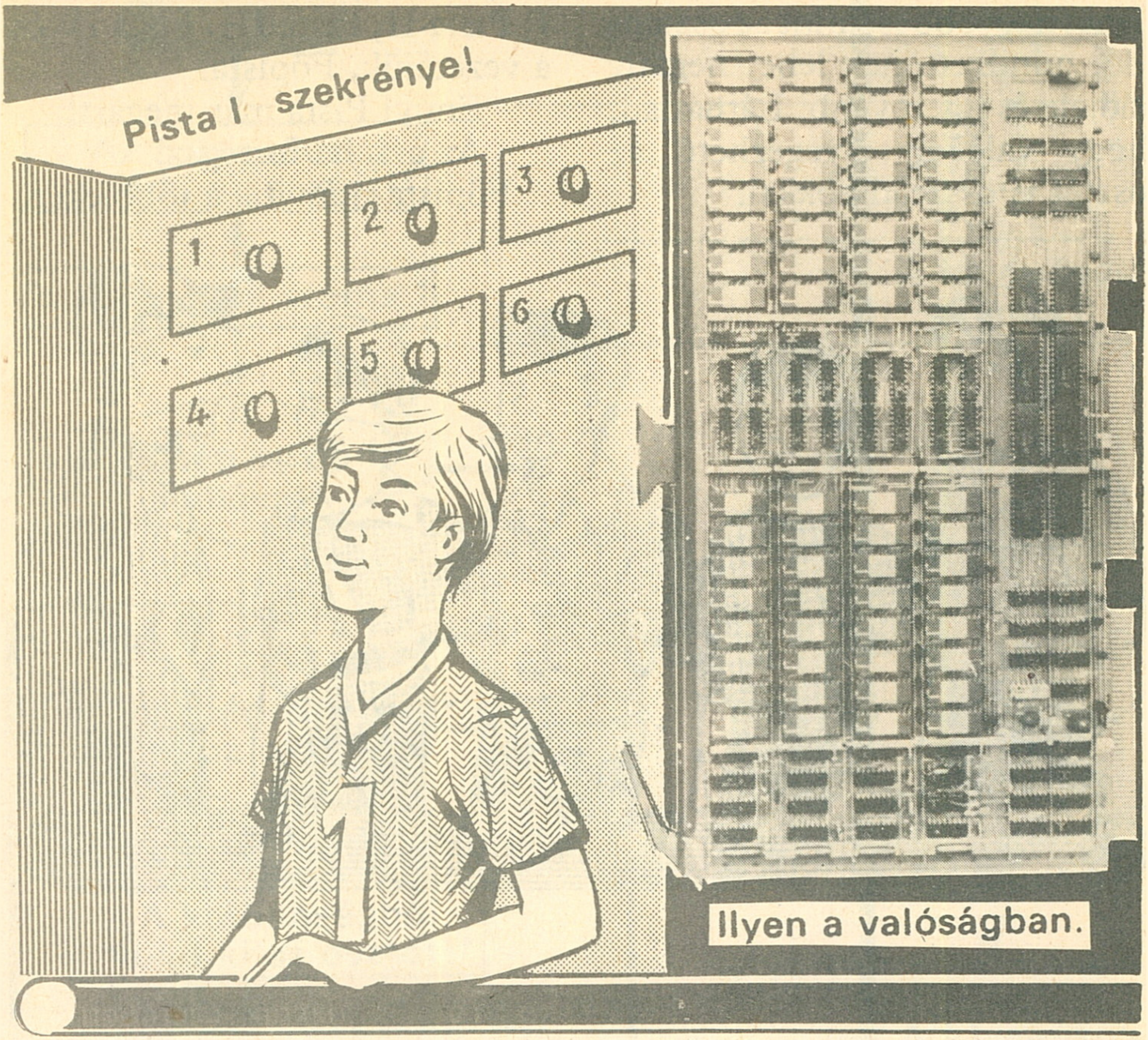
– Az árt, ha fecsegsz!

Főpista irányítja a többieket.



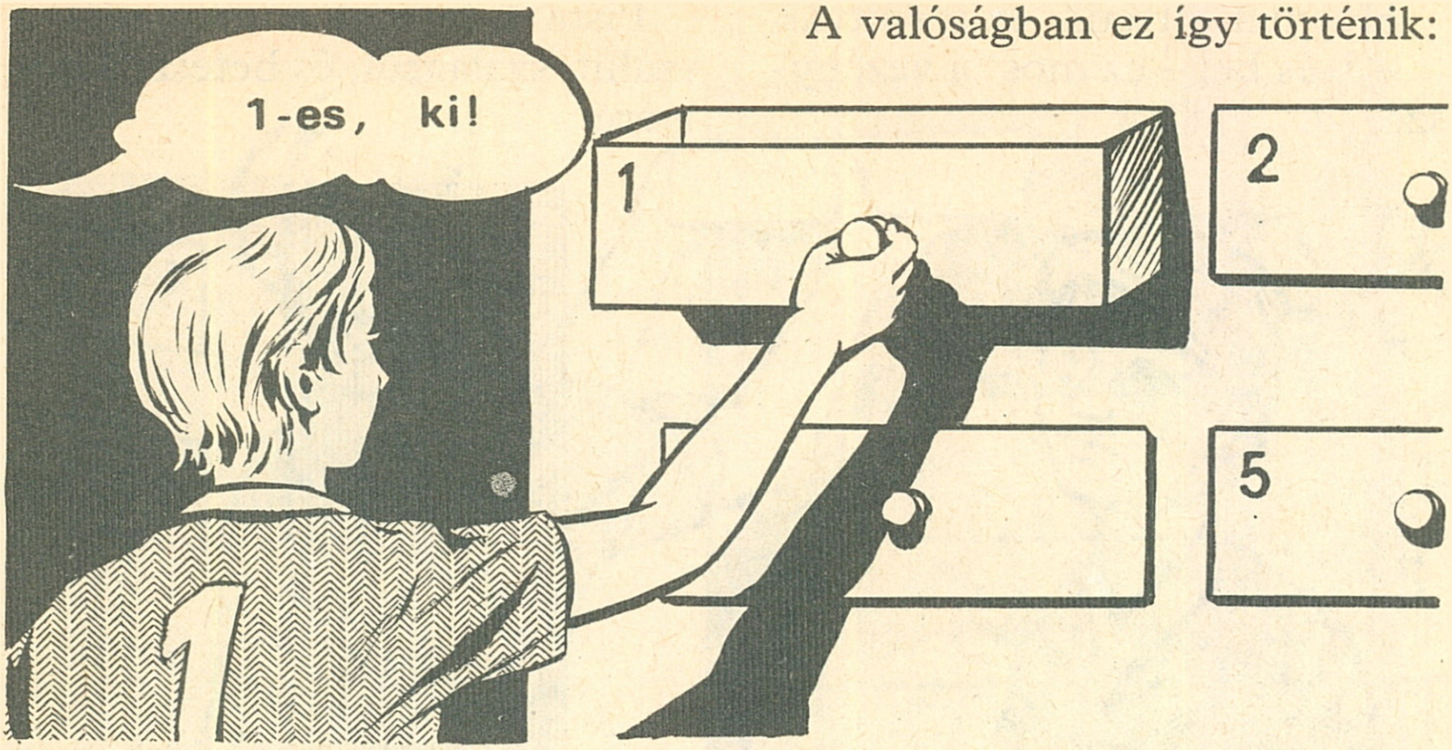
Mindenkinek megvan a saját feladata, lássuk most ezeket sorjában!

Pista I egy nagy, fiókos szekrényt kezel, amely szállítószalag mellett áll:



A szekrény fiókjai sorban meg vannak számozva: 1-es, 2-es és így tovább. E számoknak ugyanaz a szerepük, mint az irányítószámoknak a leveleken, vagyis ezek a fiókok címei. Pista I munkája abból áll, hogy megkeres egy adott című fiókot, és a tartalmát a szállítószalagra teszi, vagy fordítva: a szállítószalag által hozott csomagot beleteszi a megfelelő fiókba.

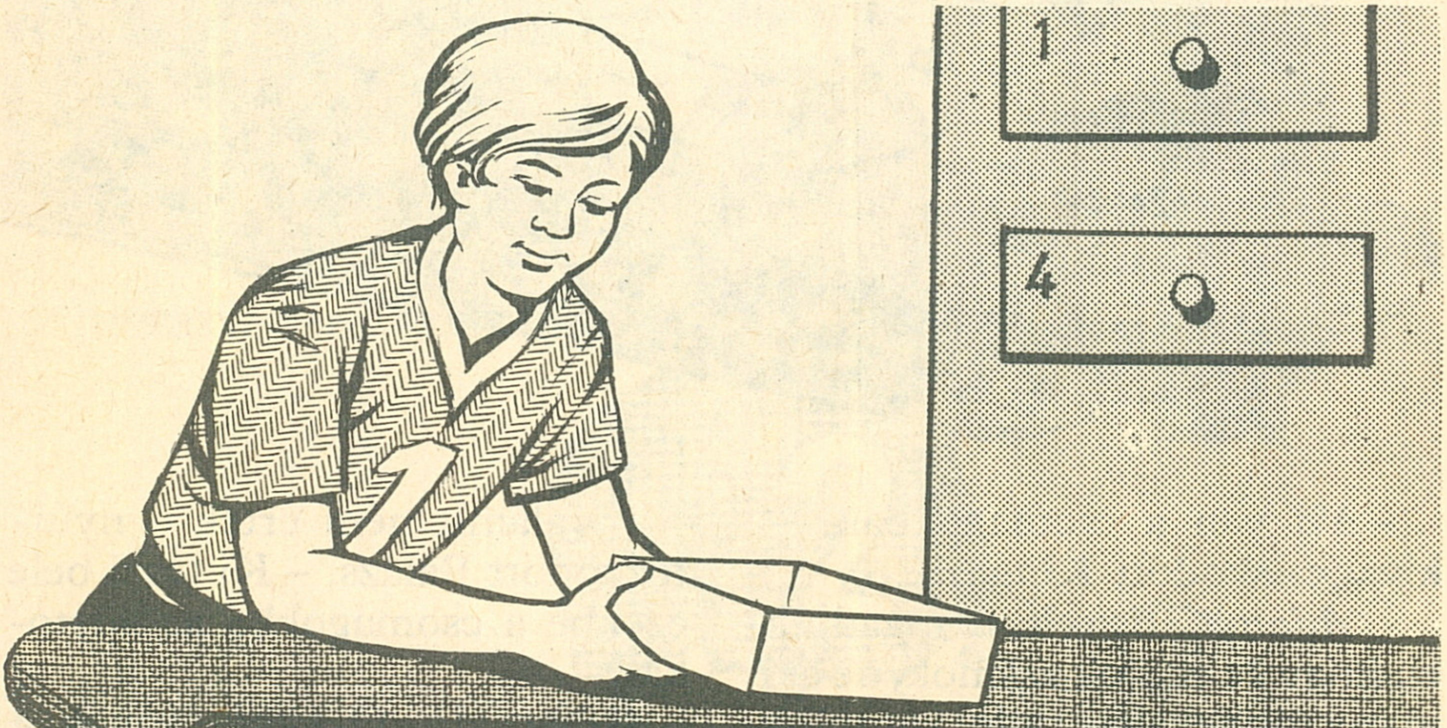
A valóságban ez így történik:



– 1-es ki! – hallja Pista I, és ezt is teszi.

Kihúzza az 1-es fiókot.

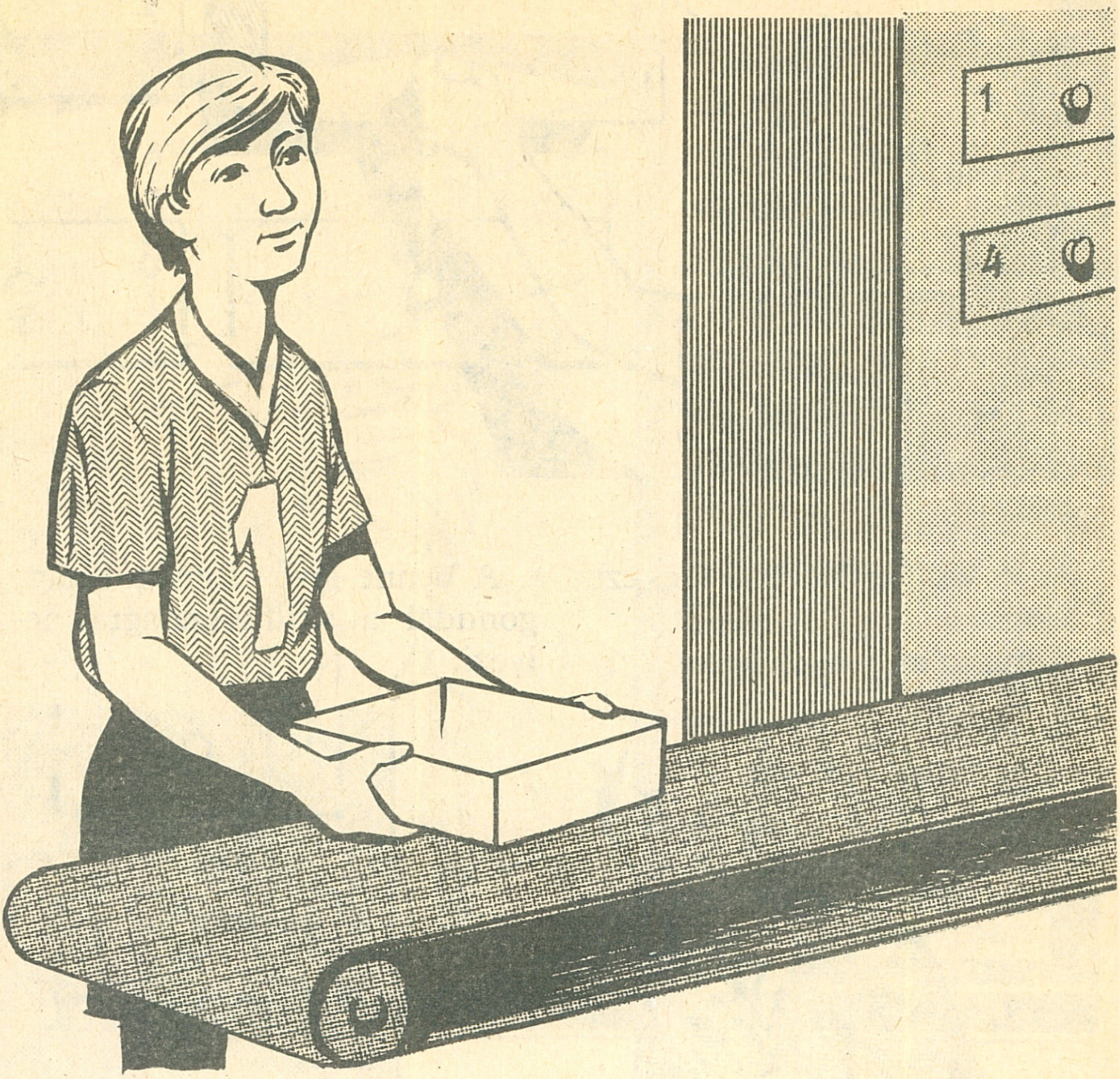
A benne lévő csomagot nagy gonddal a szállítószalagra helyezi.



Hogyan történik ez fordítva?

– 1-es be! – ez most a vezényszó.

Pista I leveszi a csomagot a szállítószalagról, és beteszi a helyére.



– Mi van a csomagokban? – kíváncsiskodott Kinga.

– *Adat*ok. Egyelőre fogadjuk el, hogy az adatok számok, de aki máris többet akar tudni róluk, az nyugodtan kukkantson bele a következő fejezetbe.

– Valamit nem értek – nyújtózkodott Balázs. – Ki teszi bele ezekbe a csomagokba az adatokat?

– Pista II. Ő kapja annak a pontos leírását, hogy mit kell csinálni.


1
2
5
olivetti
ON
S2
S1
Q W E R T Y U
A S D F G H J
Z X C V B N
2

Először elolvassa, majd a leírásnak megfelelően beleteszi a csomagokba az adatokat. A szalag pedig elviszi a csomagokat.



– Soká tart még? – kérdezte tapintatosan Csaba, a húga pedig gátlástalanul unatkozott.

Mit tehettem, gyorsan ott folytattam, ahol abbahagytam.

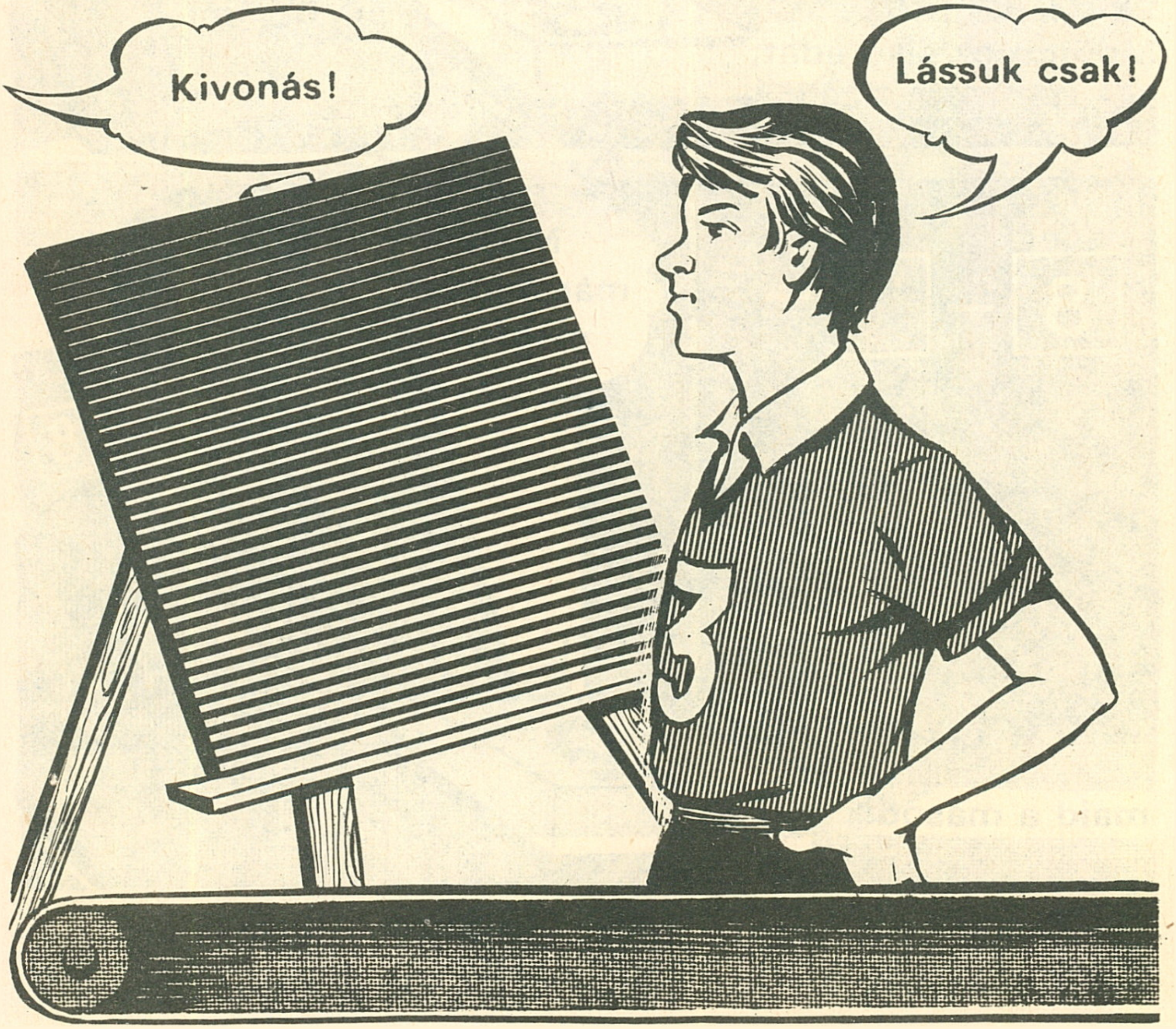
A maradék Pisták következnek

Pista III szintén a szállítószalag mellett áll, az ő dolga a számolás.

A számok csomagokban érkeznek Pista III-hoz a szállítószalagon.

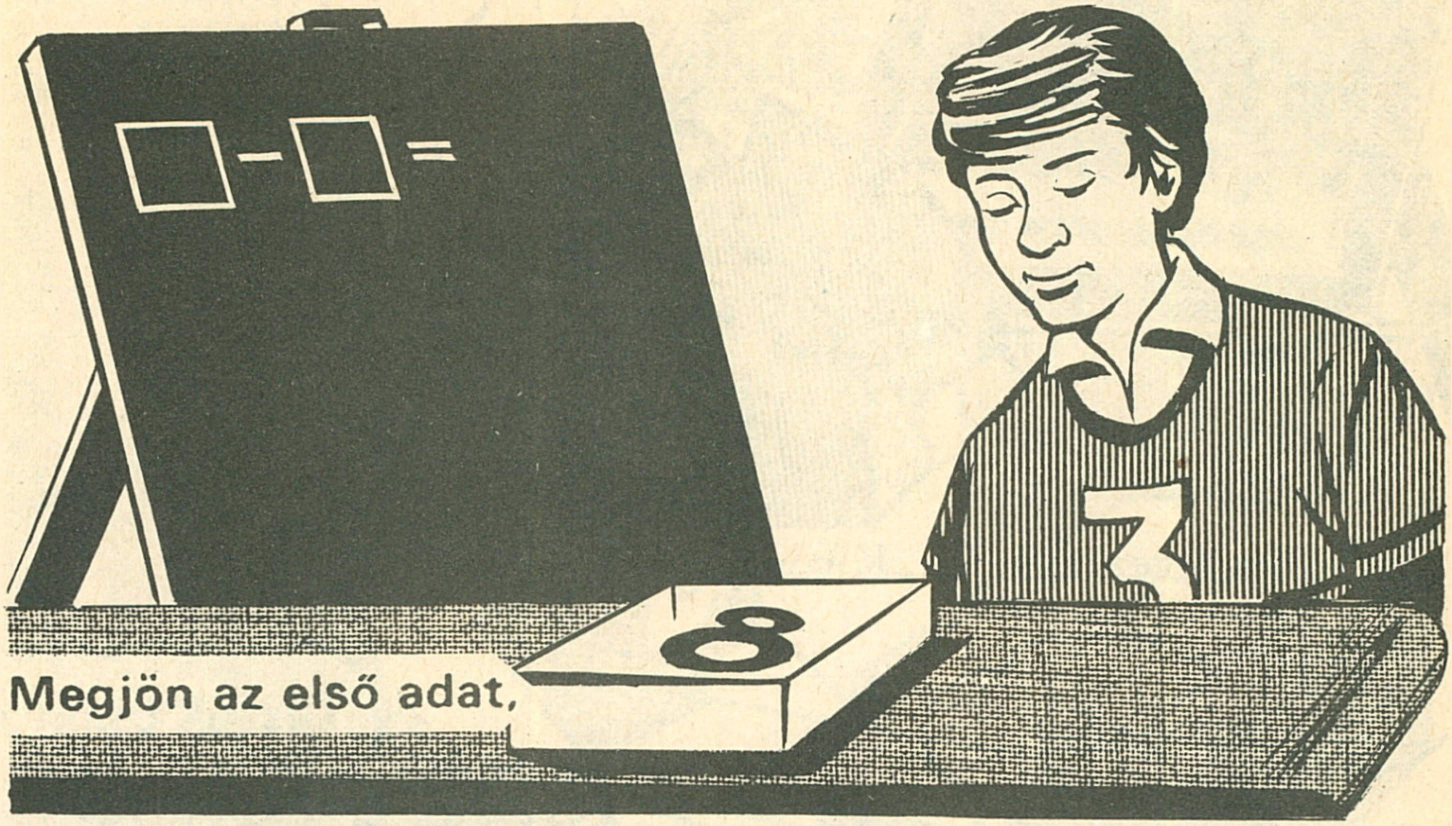


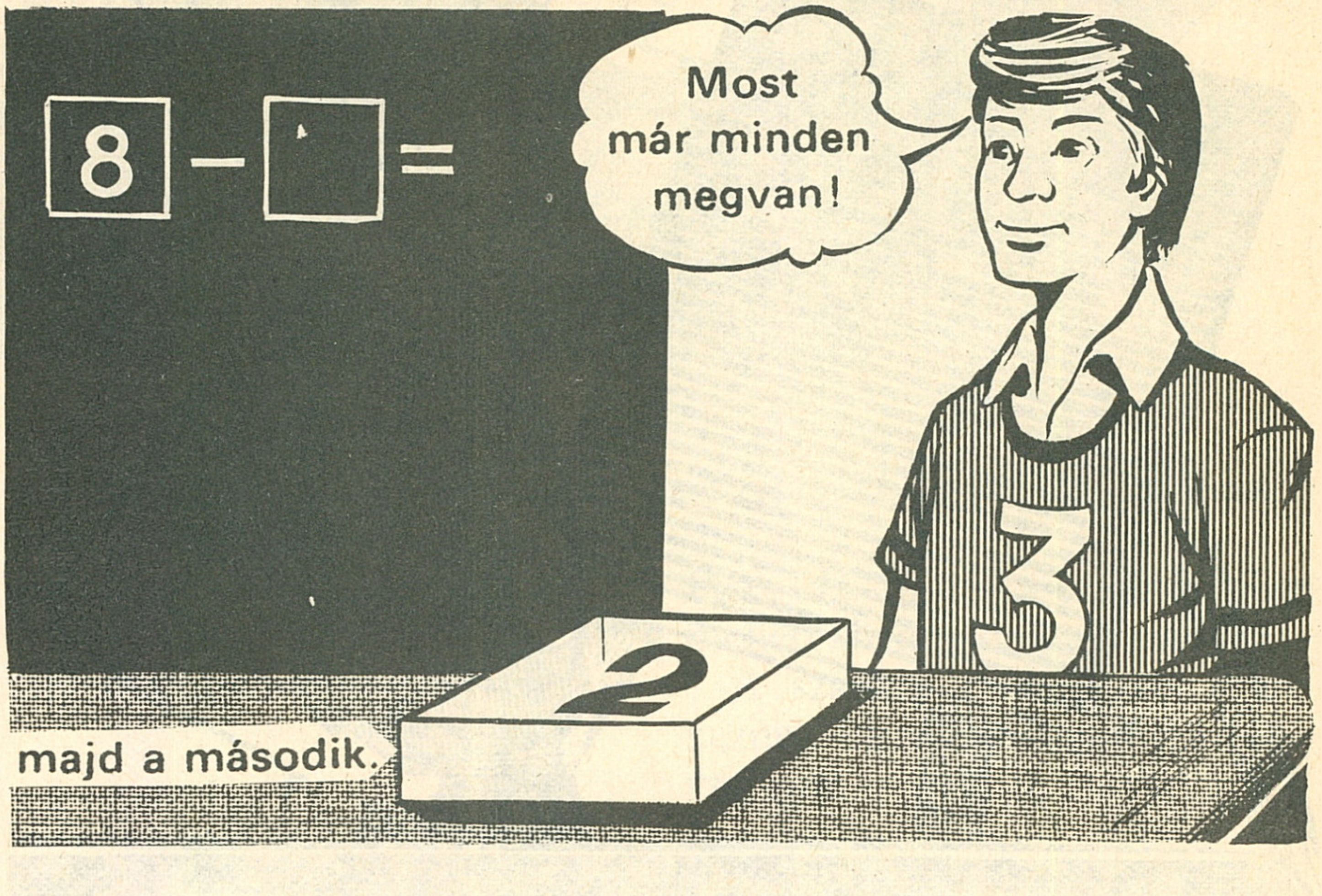
Meghallja a vezényszót: „Kivonás!”



Erre leveszi a szállítószalagról a most érkező két csomagot, a bennük lévő számokat kivonja, és az eredményt megint egy csomagban a szállítószalagra teszi.

Pontosan így:

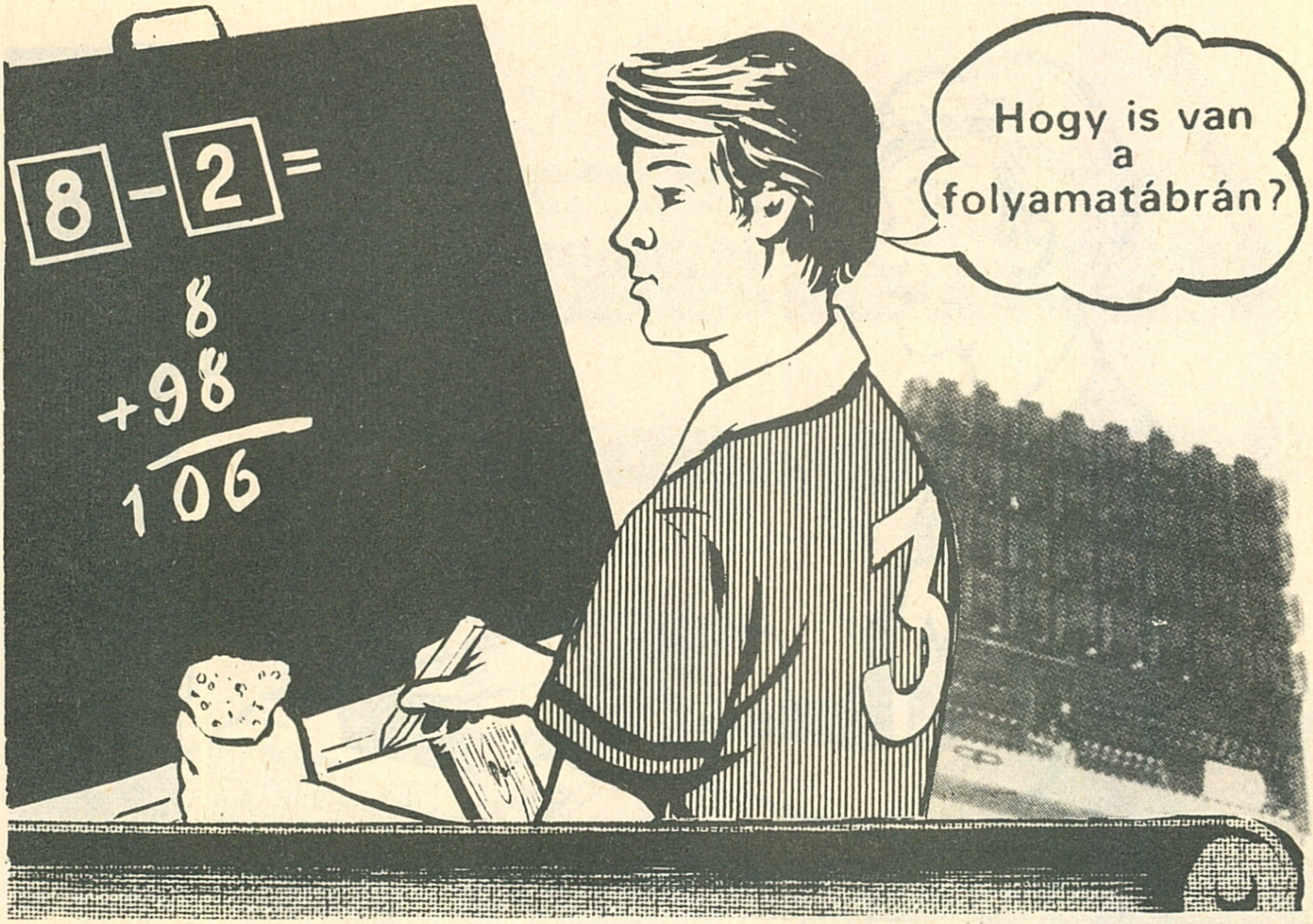


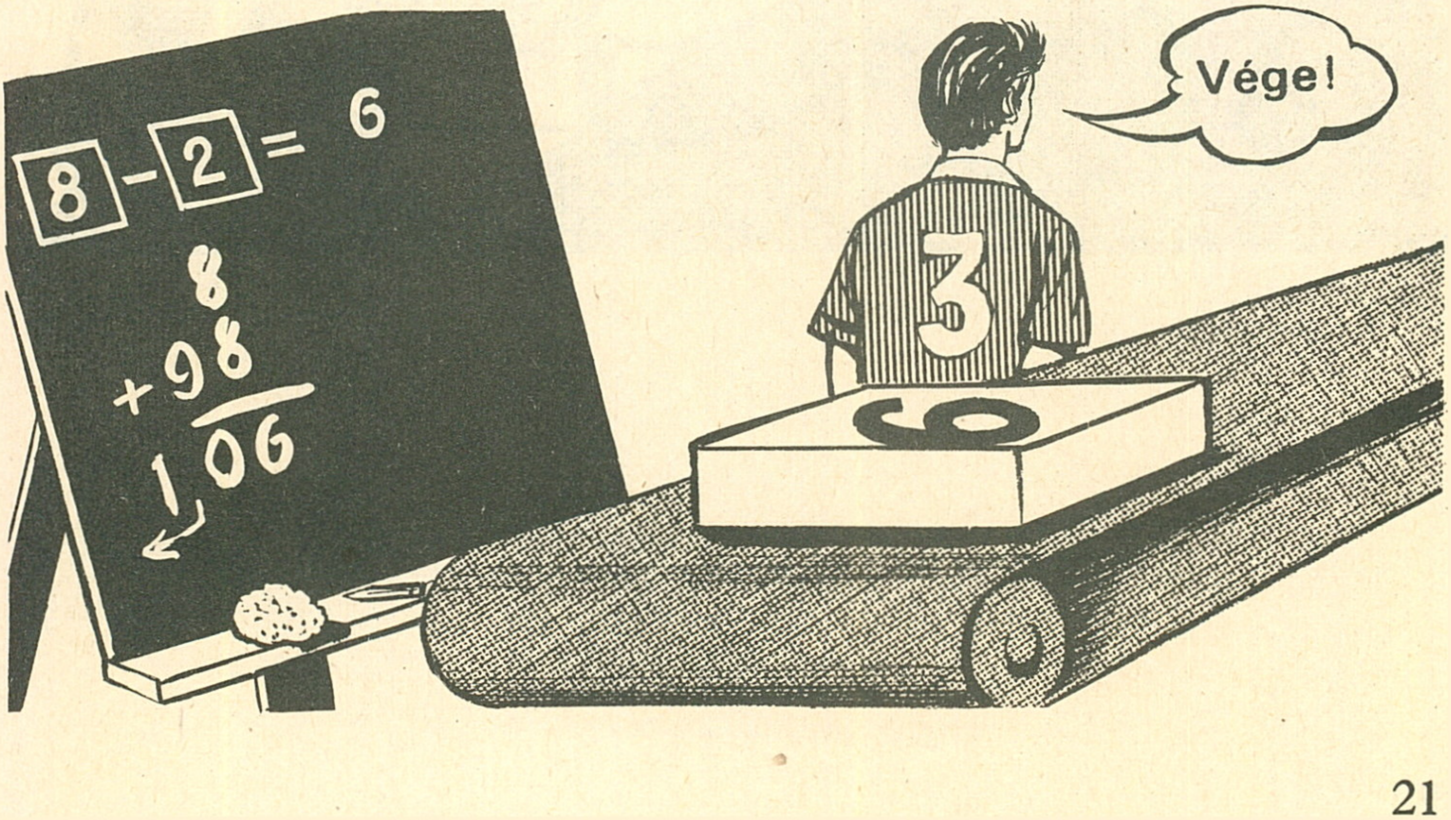


Elhangzik a vezényszó. Megtörténik a kivonás.

– Az eredmény a szállítószalagra kerül, és kész! – kottyantott bele Kinga.

– Oda kerül, de még nincs vé-



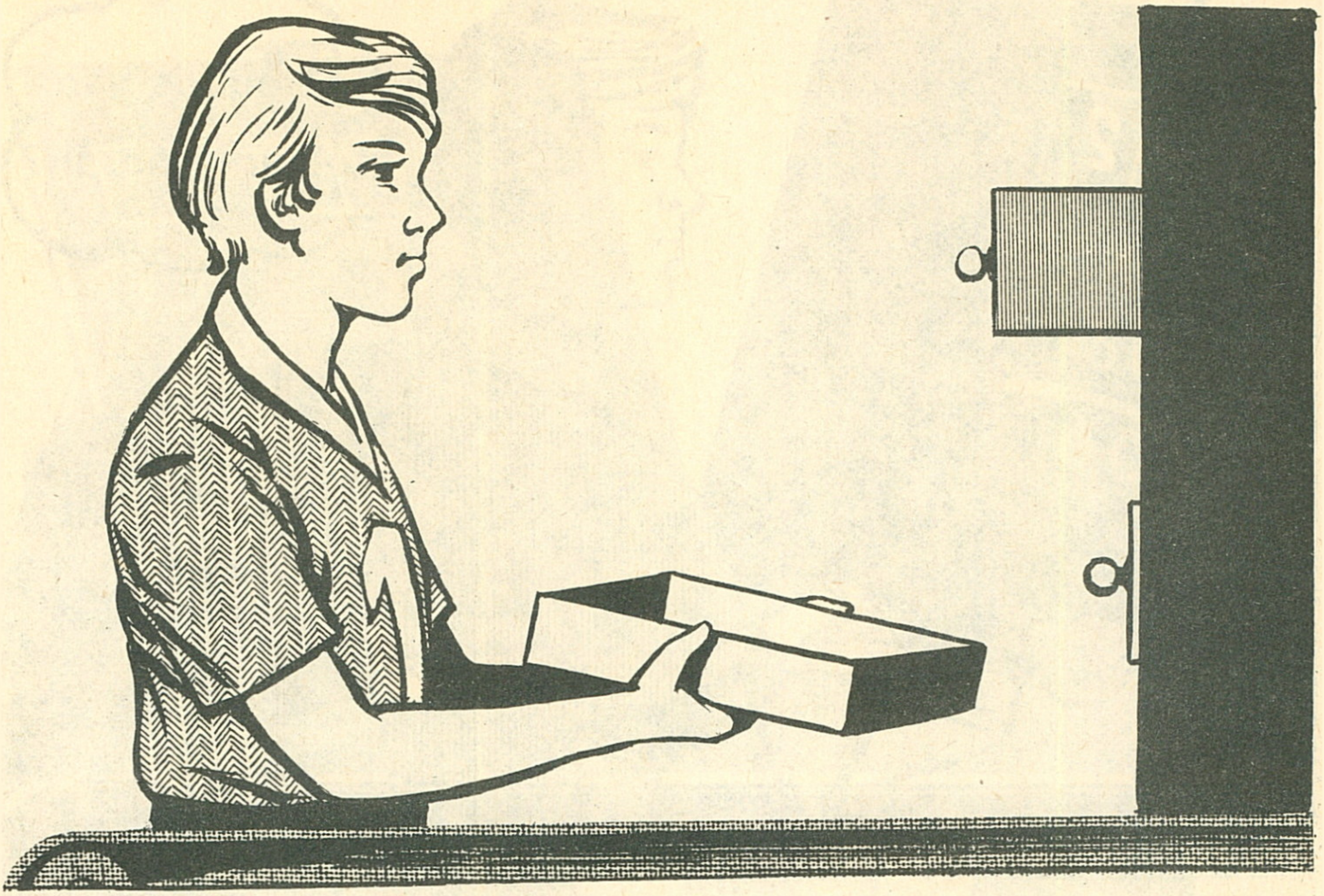


ge. Az adatok a szállítószalagon csak utaznak, tárolni azokat...

– Pista I szekrényében kell!

– Úgy van.

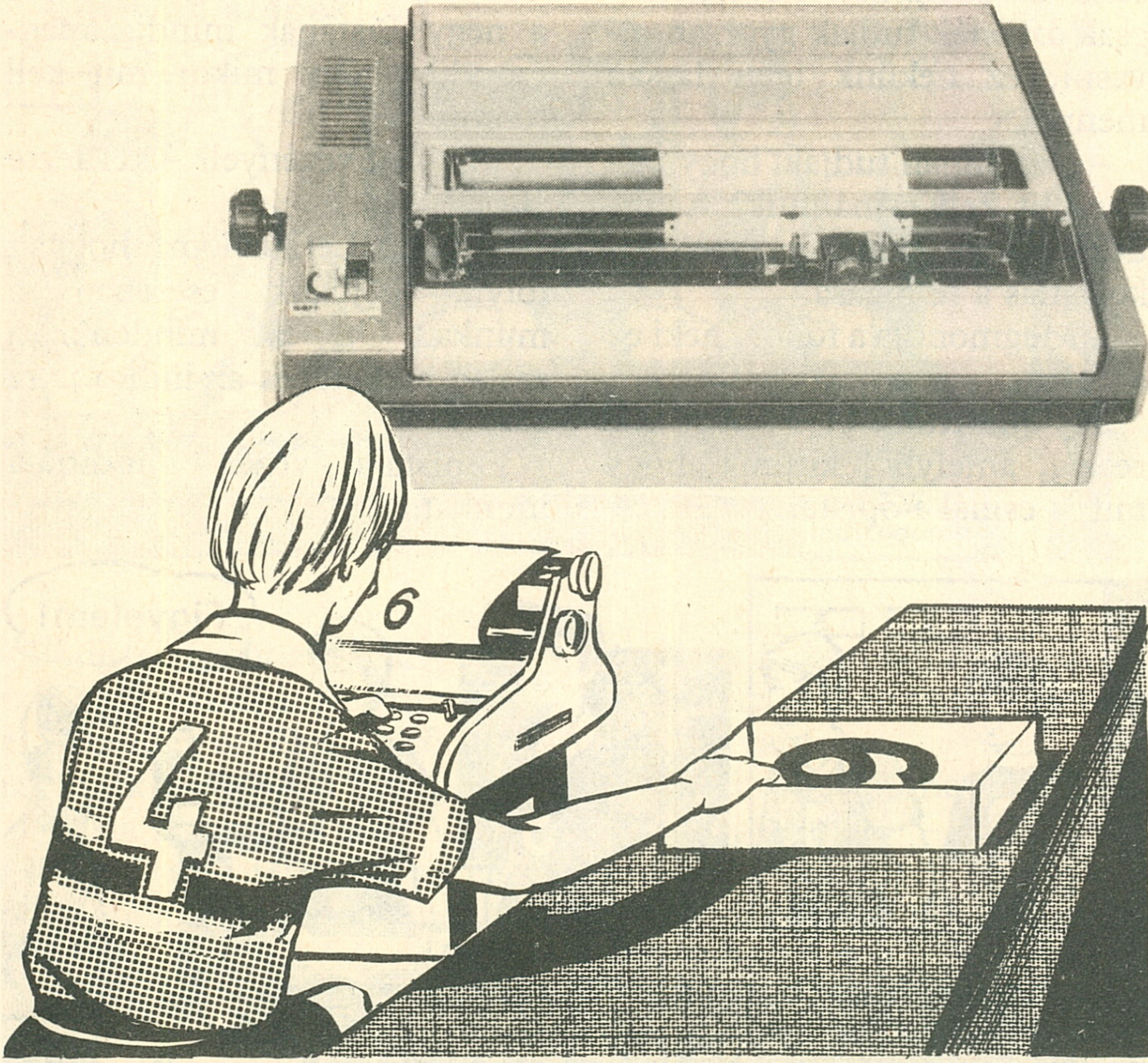
A kivonás végén az eredmény egy csomagban Pista I elé kerül, aki megfogja, és beteszi azt egy fiókba.



Pista IV-nek írógépe van.

– Ez fog gépelni? – kérdezte Balázs.

– Igen. Hozzá is a szállítószalag viszi a csomagot, és amit benne talál, azt leírja.



– Miért kell újból leírnia, hiszen a csomagban is le van írva?

– A csomagbeli írást – amelyet *belső ábrázolás*nak neveznek – csak a Pisták tudják gyorsan olvasni, ez nekünk igen lassan menne.

– És honnan tudják, hogy mikor mit kell csinálniuk? – kérdezte Kinga, akinek nagyon gyakorlatias a felfogása.

– Megmondja a főnök, neki ez a dolga.

Így jutottunk el a következő részig, amelyből kitűnik, hogy mit is csinál Főpista.

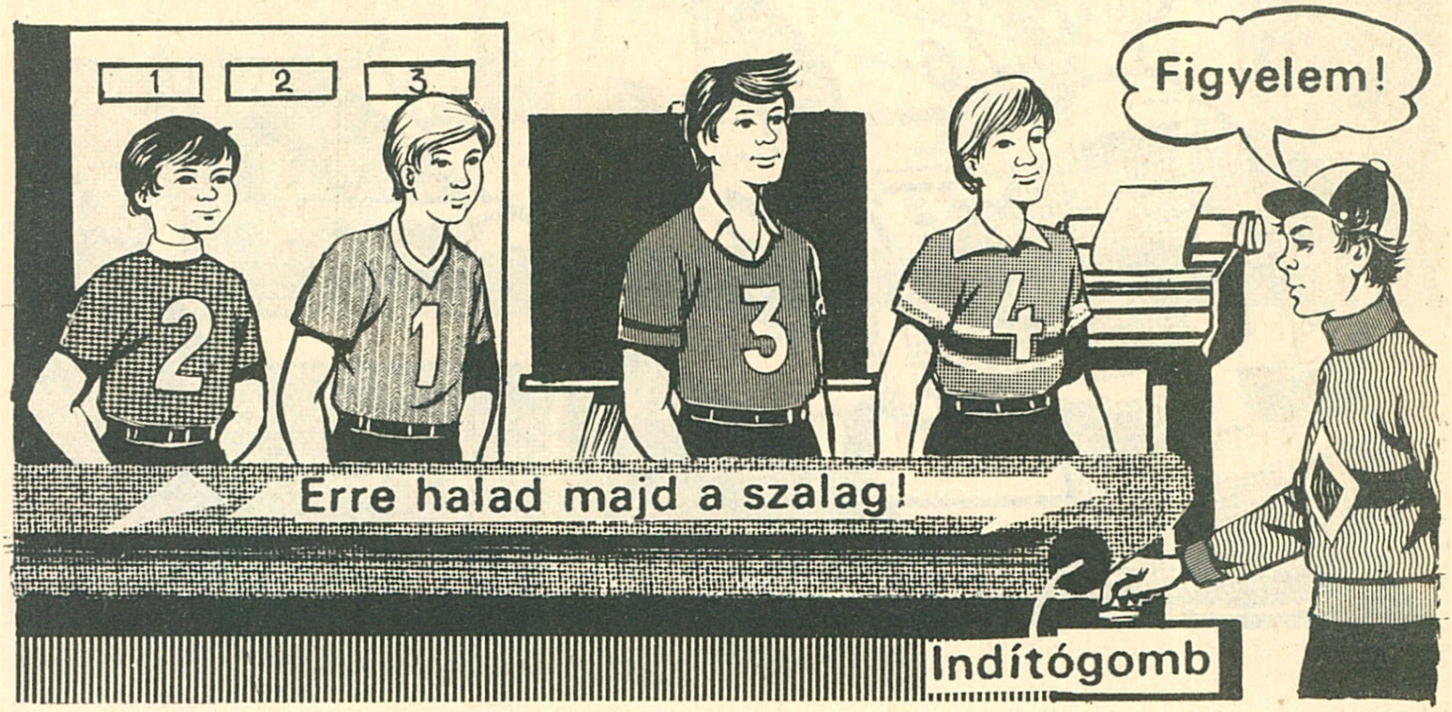
Mit is csinál Főpista?

Nos, Főpista valóban főnök. Ott áll a szállítószalag mellett, és a négy Pistának mindig megmondja, hogy mikor mit kell tennie.

– Szóval vezényel? – kérdezte Balázs.

– Igen. Lerajzolom, hogyan folyik példánk esetében a munka. Ott áll mindenki a szalag mellett, és az indító jelre vár.

Főpista a – vezér – elindítja a munkát:



– Pista II, olvass!

Erre Pista II kézbe veszi a leírást, elolvassa, és három csomagot helyez a szállítószalagra.



Főpista elindítja a szalagot: a csomagok közelednek Pista I felé. Épp hogy odaért az első, már hallatszik is a következő utasítás:

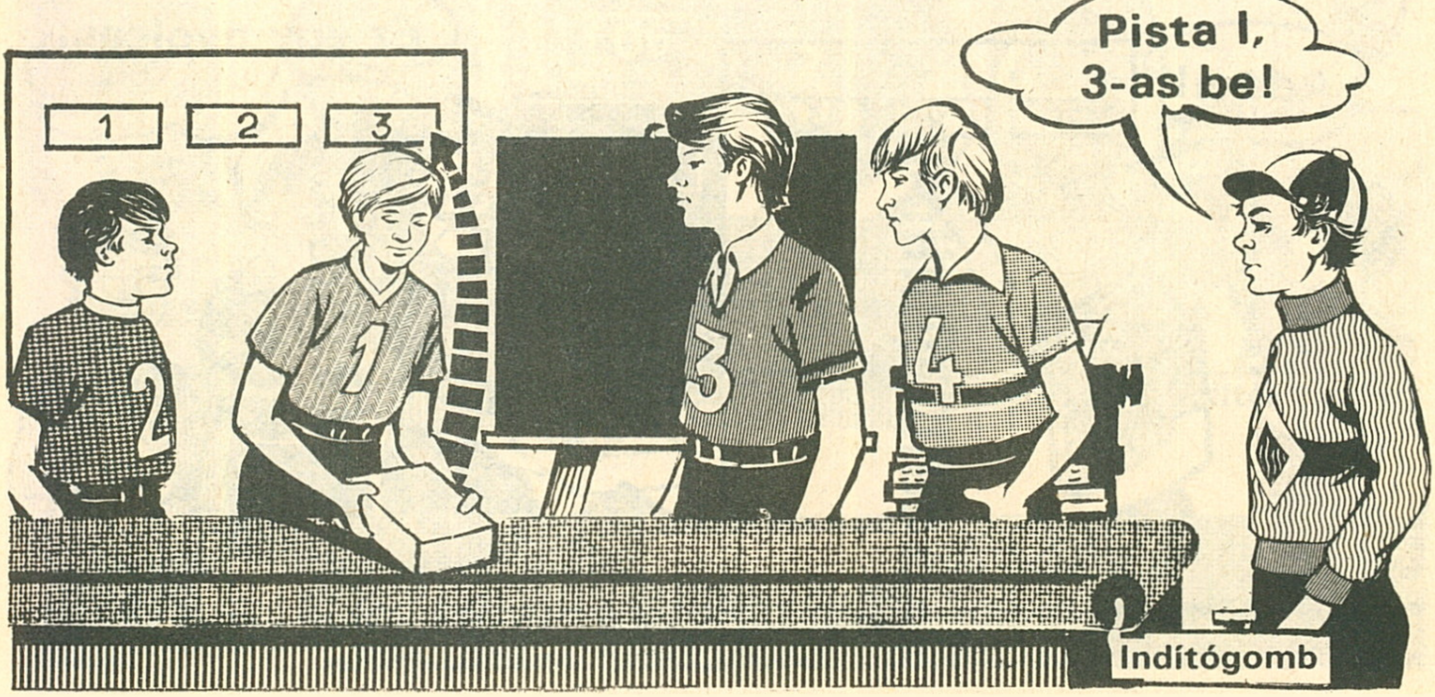
– Pista I, 1-es be!

Pista I alig tette be az első csomagot az 1-es fiókba, már ott is van az új csomag és a parancs:

– Pista I, 2-es be! Mire ezt is teljesítette, a harmadik csomag is előtte van.

– Ezt biztosan a harmadik fiókba fogja tenni – vélekedett Kinga.

– Eltaláltad – mondtam, és felrajzoltam ezt a helyzetet.

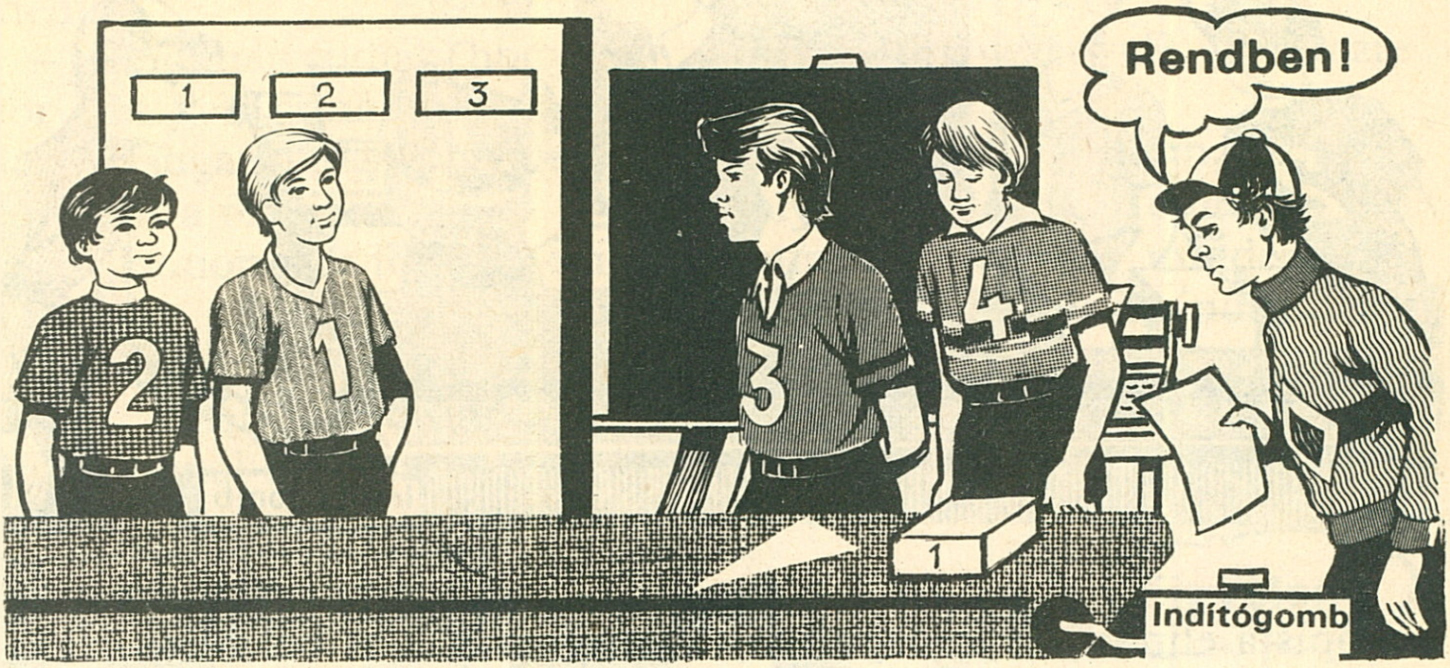


Ami most következik, azt úgysem találná ki senki:

– Pista I, 1-es ki! – vezényel Főpista.

Az 1-es fiókban lévő csomag a szállítószalagon Főpista elé kerül, aki kibontja, és elolvassa a tartalmát:

„A 2-es fiók tartalmából ki kell vonni a 3-as fiók tartalmát, s az eredményt – természetesen – le kell írni.”

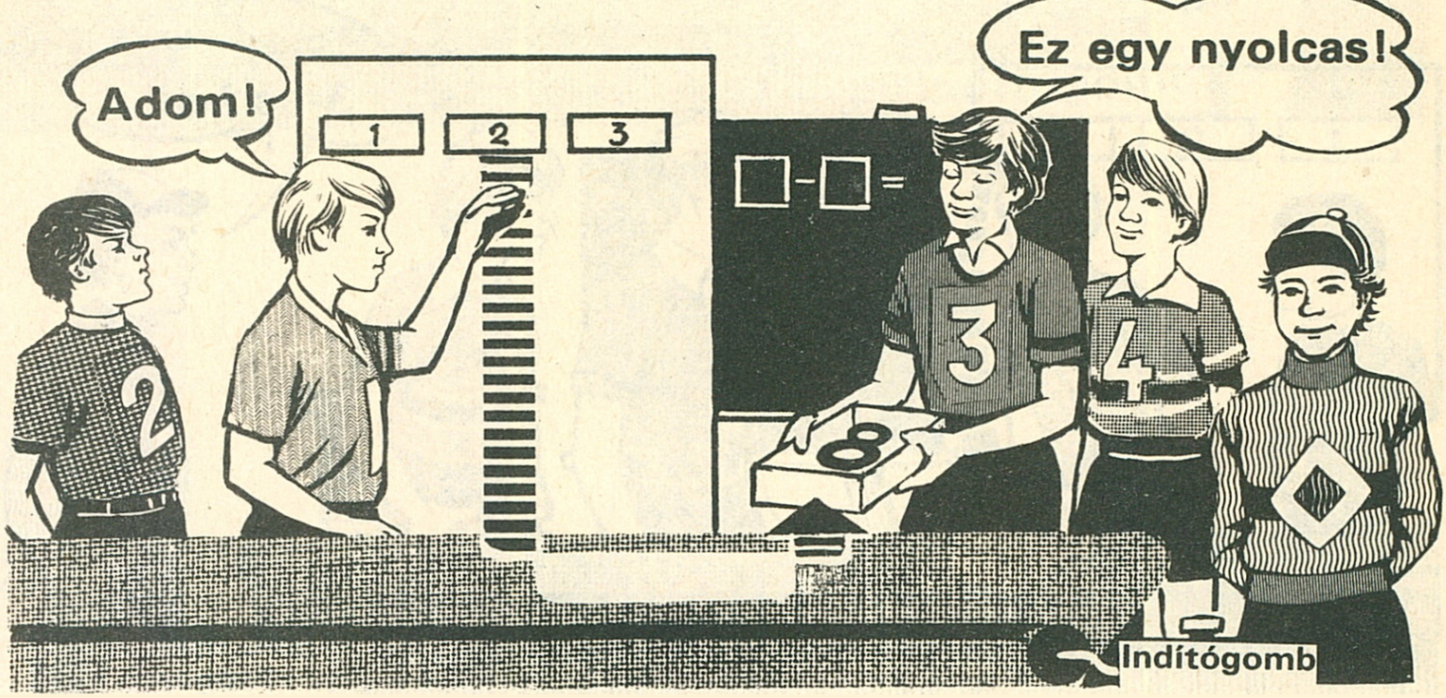


Innen már minden úgy megy, mint a karikacsapás.

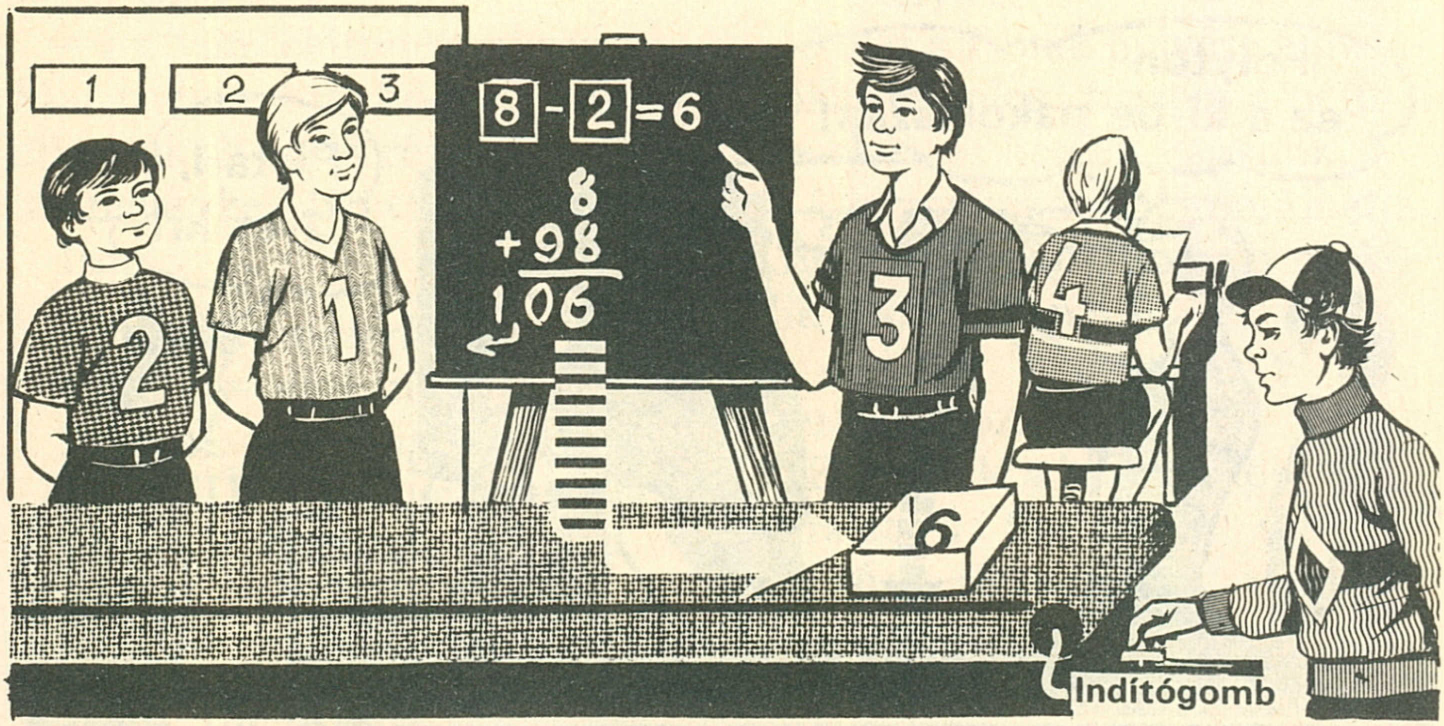
– Pista III, kivonás! – hallatszik, és Pista III készülődni kezd.

– Pista I, 2-es ki! – szól újra Főpista, és a csomag már utazik is Pista III felé. Amikor Pista III levette a csomagot, ismét Főpista hangja hallatszik:

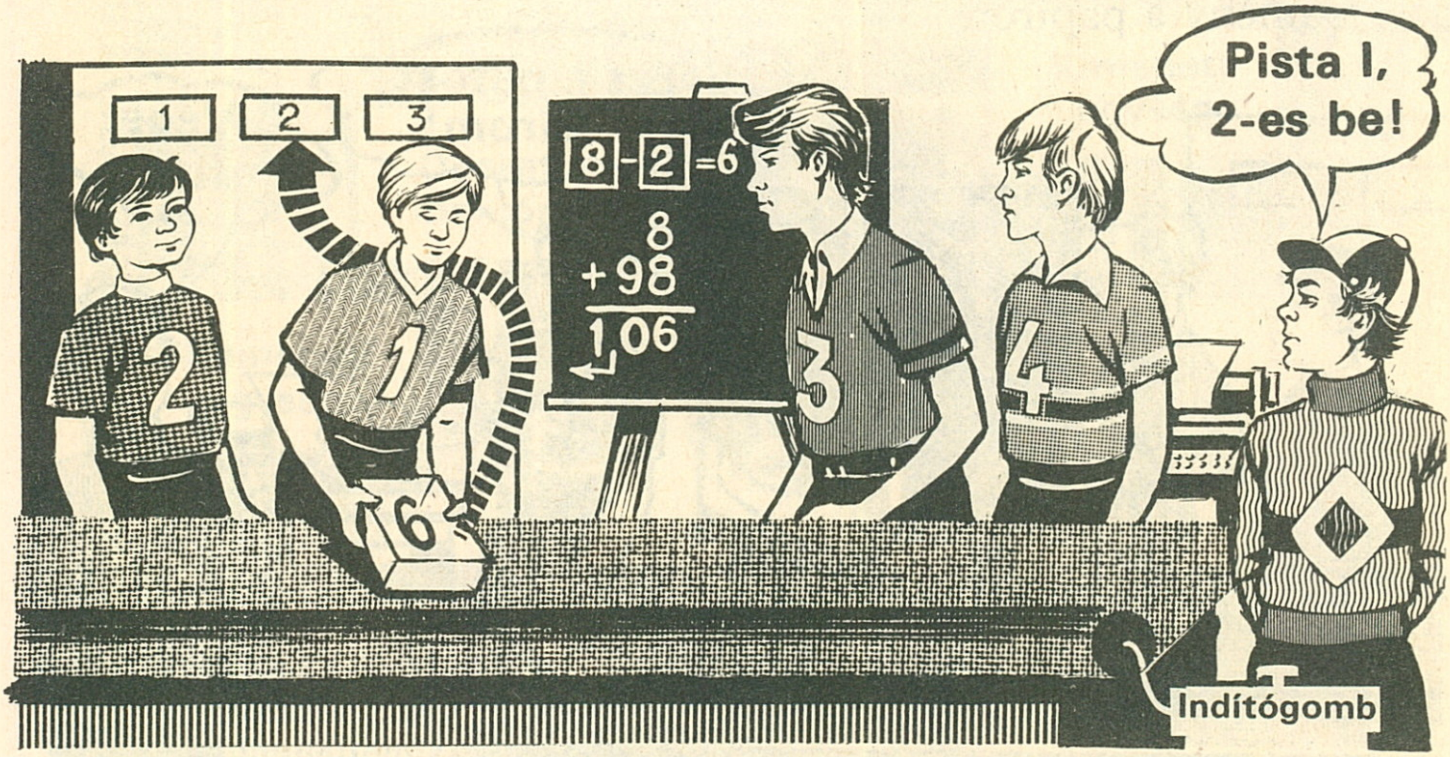
– Pista I, 3-as ki!



Most már Pista III dolgozik: elvégzi a kivonást, és az eredményt a szalagra teszi.



– Pista I, 2-es be! – intézkedik Főpista, és az eredmény ezzel a szekrénybe kerül.



A munkát Pista IV fejezi be.
Először a szállítószalag elviszi hozzá az adatot:



– Pista IV, gépelj! – utasítja Főpista, s így jelenik meg végül az eredmény a papíron.



– Apu, mondd meg végre, hol az a *számítógép*! – kezdett nyekeregni Balázs.

– Itt van előttetek a rajzon. Már tudjátok is, hogyan működik, csak az egyes részeinek pontos nevét kell behelyettesíteni.

Íme:

Főpista = *vezérlőegység*,
Pista I = *operatív memória*,
Pista II = *bemeneti egység*,
Pista III = *feldolgozóegység*,
Pista IV = *kimeneti egység*.

– És a szállítószalaggal mi lesz? – aggályoskodott Kinga.

– A szállítószalag neve *adatsín*.

– Ennyire egyszerű? – hitetlenkedett Csaba.

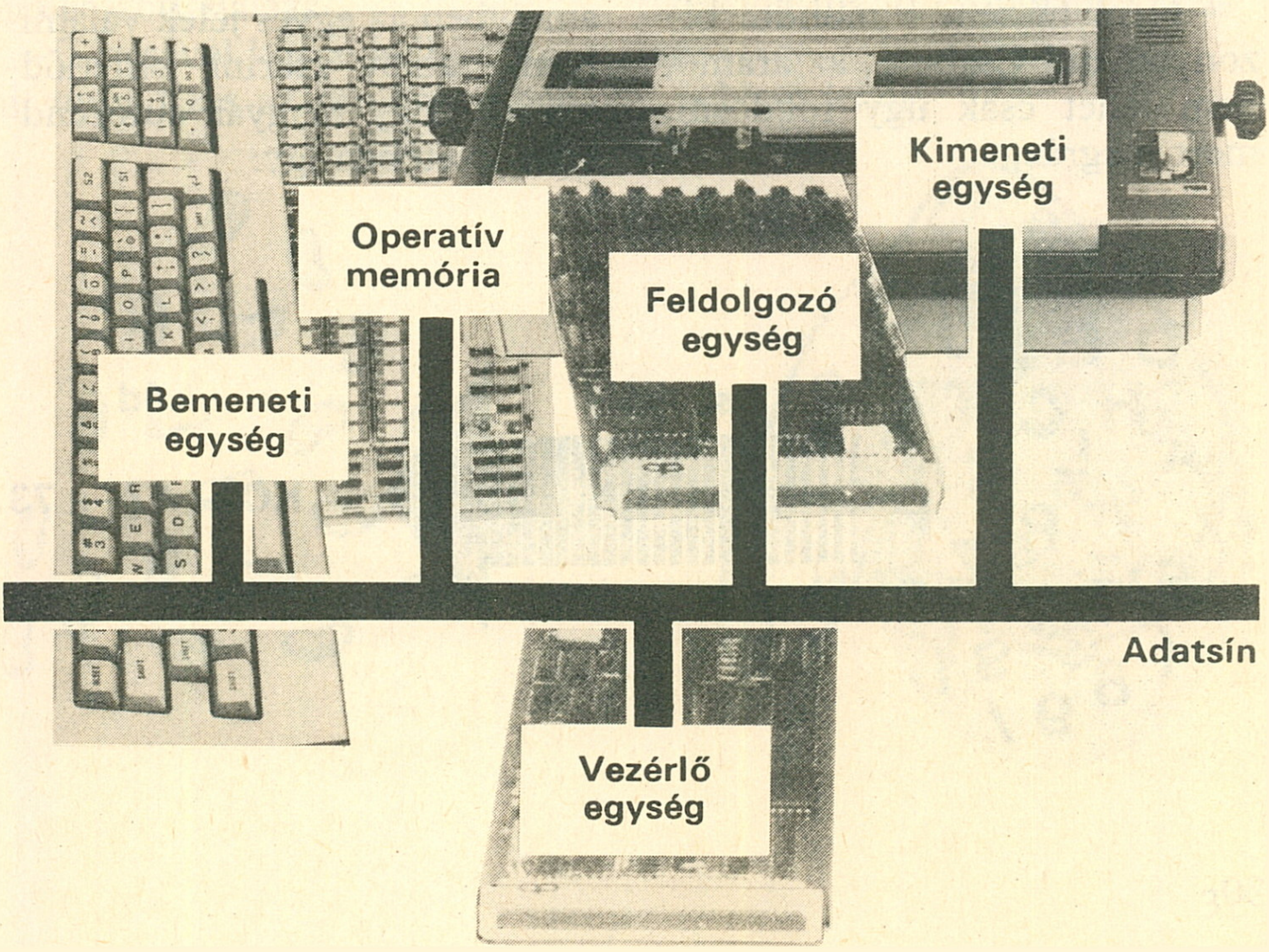
– Ennyire.

Most a Pisták helyére téglalapokat rajzolok, beírom a pontos neveket, és készen vagyunk.

Akármelyikőtök elmondhatja: „Én már tudom, hogyan működik a számítógép!"

GONDOLKOZZATOK!

1. Rajzold le a kivonás folyamatábráját a fejezetben szereplő 8 – 16 = – 8 esetre!
2. Tudnál olyan folyamatábrát rajzolni, amely a 41 – 12 = 29 és 26 – 34 = – 8 esetekre egyaránt jó?



Mi van a dobozban?

– A számokat miből csinálják? – érdeklődött Balázs, mert a barkácsolás az (egyik) szenvedélye.

– Ráírják egy darab papírra, és kész! – hangzott Kinga válasza.

– Meg fogtok lepődni, ha meghalljátok: a számok áramból vannak!

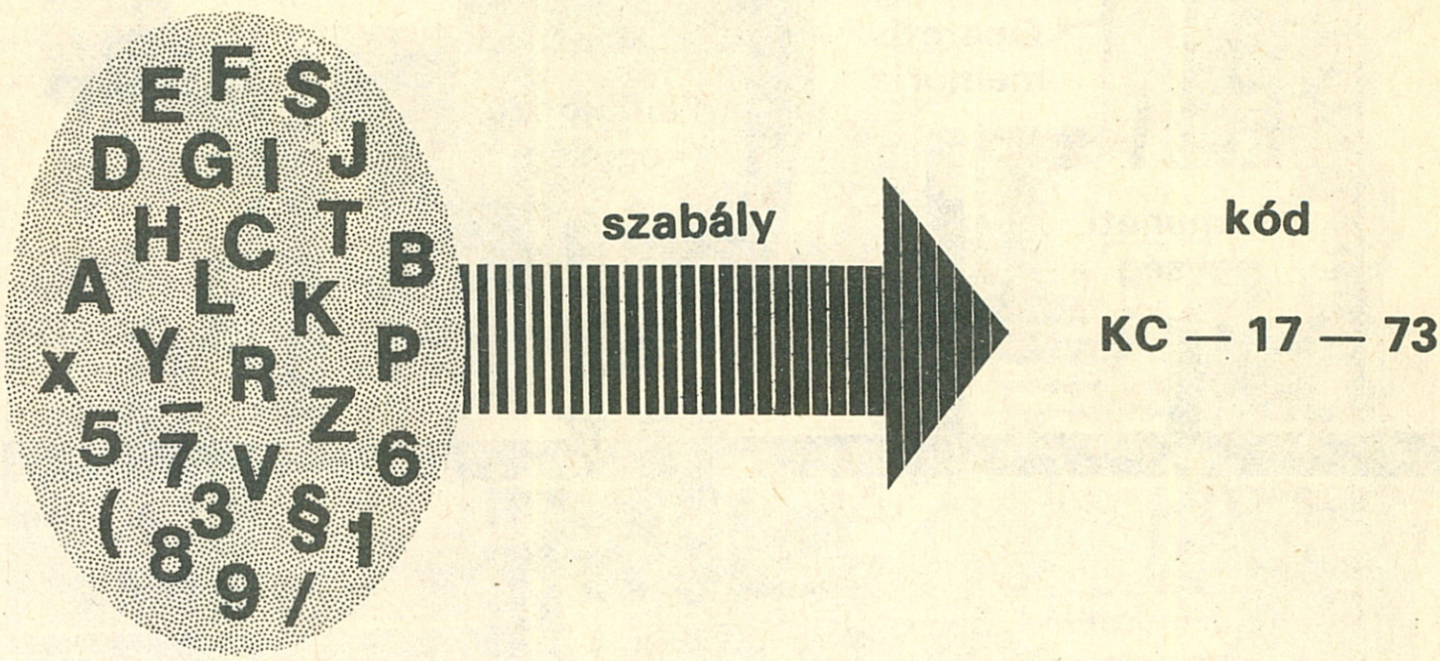
Mindenki önkéntelenül a legközelebbi konnektorra nézett, de az éppen olyan volt, mint eddig.

– Nem hiszem – mondta bizonytalanul Csaba –, az áramot nem lehet csak úgy fogdosni, mert megráz.

– Pedig mégiscsak ez a valóság, és nem is kell hozzá megfogni. Figyeljetek csak ide! Ki tudja, mi az a *kód?*

– Szám- vagy betűcsoport, amitől kinyílik a páncélszekrény. Egyszer láttam egy filmet, abban is... – lendült bele Kinga, de megelőztem:

– Az is kód volt, de ti is ismertek kódokat: kód egy autórendszám, sőt egy tízes számrendszerben felírt szám is az. A kód ugyanis meghatározott jelkészletből előállított jelsorozat, amelyben az egyes jelek valamilyen szabály szerint ismétlődhetnek. Egy magyar autórendszám esetében így:

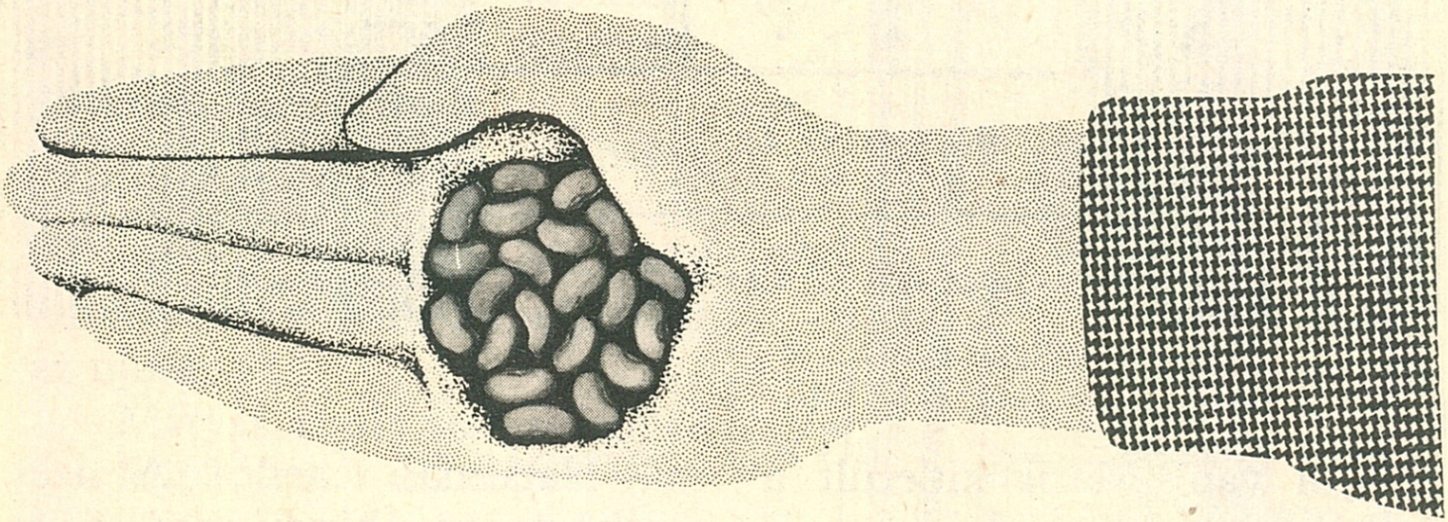


A tízes számrendszerben a jelkészlet tíz jelből, a 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 számjegyekből áll.

– A szabály a helyiérték lesz?

– Úgy van. Bármit is számolunk, mindig tízesével csoportosítjuk a tárgyakat.

– Itt van ez a marék bab.

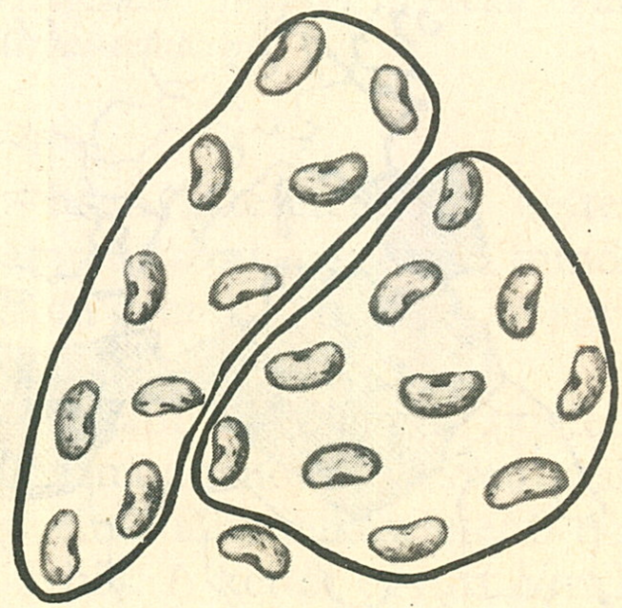


– Ezek inkább csipetkék, de majd én megszámolom őket! – ajánlkozott Balázs.

– Tízesével körbe kell keríteni, mert tízes számrendszerben dolgozunk, azután megnézni, hogy a tízes csoportokból lehet-e tízesével újabb csoportot készíteni…

– Nem lehet!





– Írd már le, hogy 21! – segítettek a többiek, de Balázs éppen úgy akarta csinálni, ahogy az iskolában nemrég megtanulta.

– Itt van! – tette le a ceruzát diadalmasan.

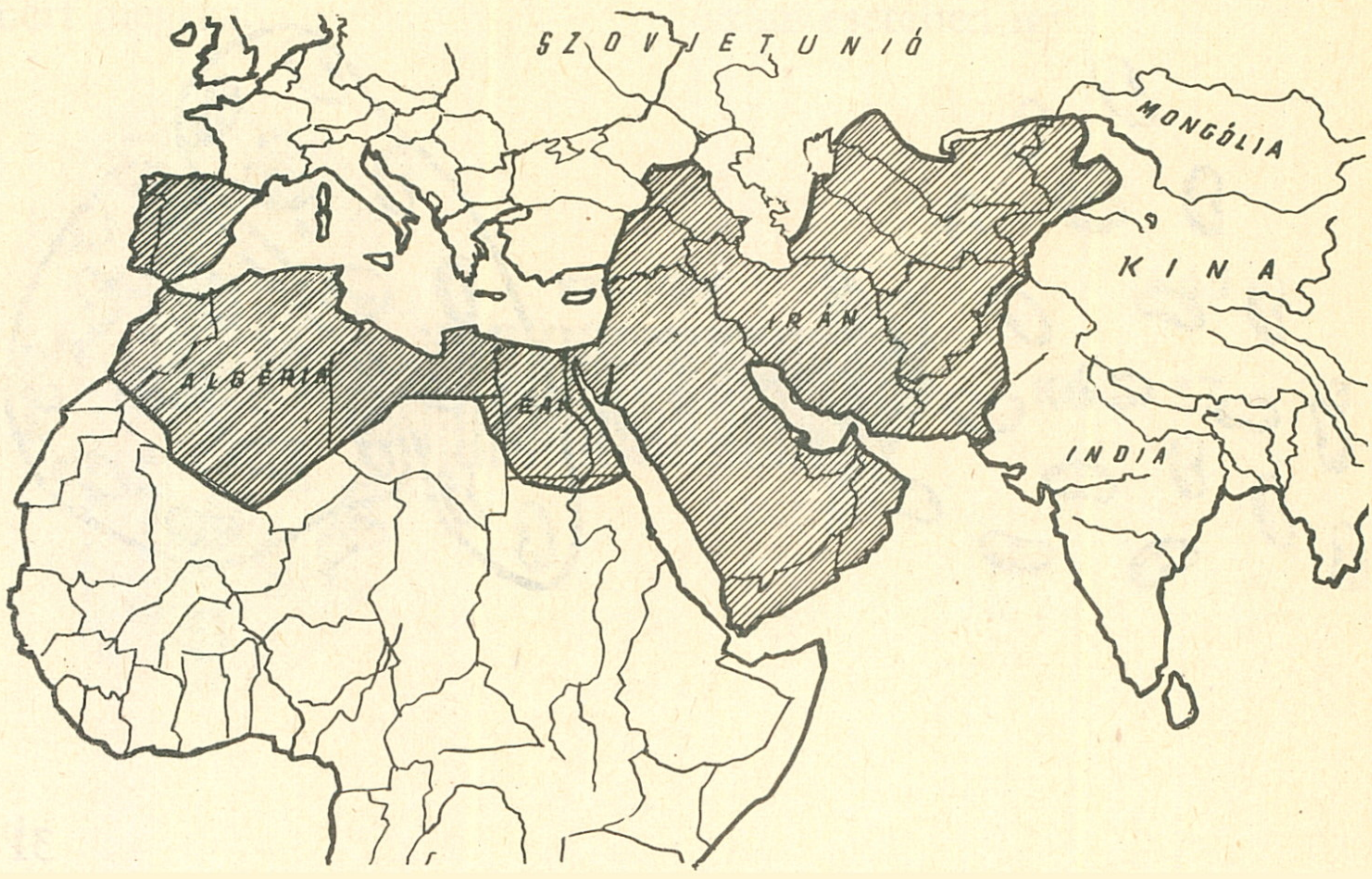
100	10	1
0	2	1
0·100+	2·10+	1·1 = 21

– Jól van! Máris kiderült a szabály: jobbról az első jegy (1) az egyeseket, a második jegy (2) a tízeseket, a harmadik jegy (0) a százasokat stb. jelenti.

– Ezt ezer éve mindenki tudja! – nyelvelt Kinga csupán jóindulatból.

– Hatalmas tévedés! A tízes számrendszer hindu eredetű, és i.sz. 500 körül alakulhatott ki. Valószínűleg arab kereskedők segítségével terjedt el nyugaton.

Térképvázlat az egykori arab birodalomról



A francia Gerbert szerzetes volt az első európai tudós, aki bizonyíthatóan használta és tanította az új számírást. Ez a szerzetes 999-ben II. Szilveszter néven pápa lett, és ő küldött koronát István királyunk megkoronázásához 1000-ben.

A számolni tudó emberek azonban a római számírást ismerték, és még vagy 500 évig tartott, míg általánosan elterjedt az újdonság.

– Mátyás király sem ismerte?

– Szerintem ismerte. Mai arab számjegyekkel van felírva az évszám egy 1485-ből való országcímeren a Mátyás-templomban.

– Ilyen kacskaringós volt a négyes?

– Igen, mert abban az időben kétféle alakban is írták ezt a számot:

 és 4



Országcímer 1485-ből (Budai Vár, Mátyás-templom)

– Ezeket értem, apa, de hogyan kerül ide az áram? – kérdezte Csaba.

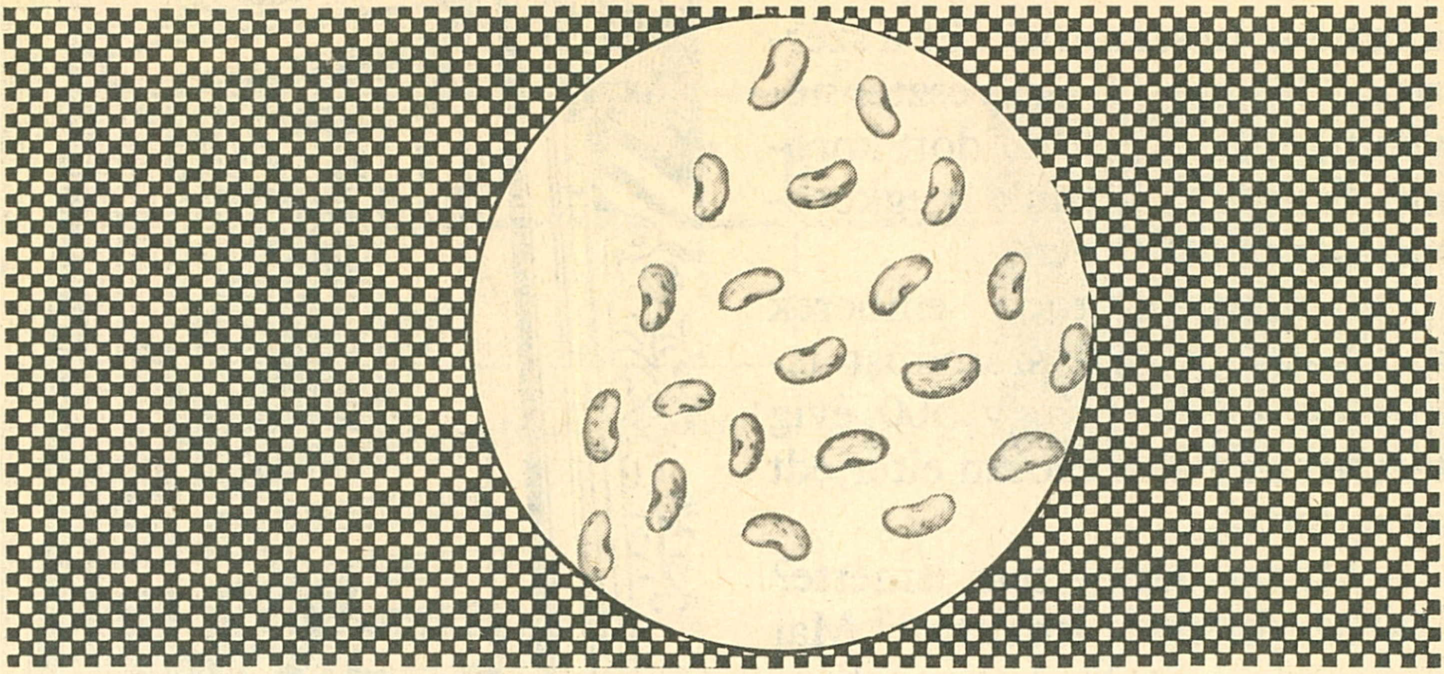
– Az áram a fény sebességével – egy másodperc alatt 300 000 km – halad a vezetékben. Ha most a számok kódolására tízes helyett *kettes számrendszert* alkalmazunk, akkor egy végtelenül egyszerűen megvalósítható adatsínhez jutunk.

– Nem szeretem a kettes számrendszert – nyafogta mindhárom gyerek –, túlságosan egyhangú!

– Inkább kéthangút kellett volna mondanotok, hiszen csak két jelből áll a jelkészlete: 0-ból és 1-ből. A kettőt együtt *bit*nek hívják. Az viszont igaz, hogy ugyanazt a számot kettes számrendszerben több jellel tudjuk

csak leírni, mint mondjuk a tízesben.

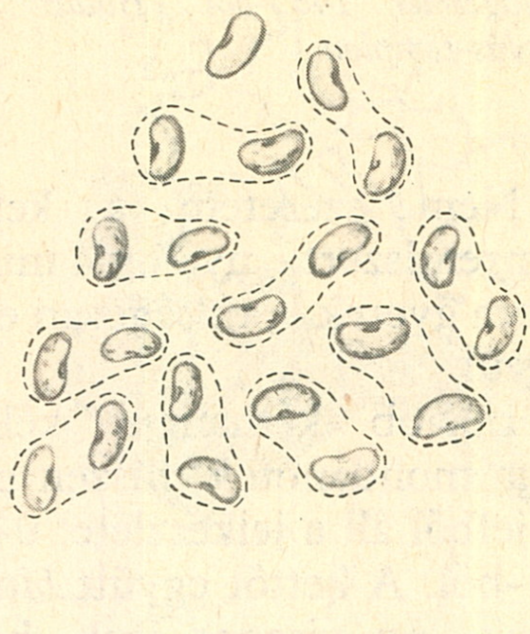
Legyenek itt megint a babjaink vagy csipetkéink:

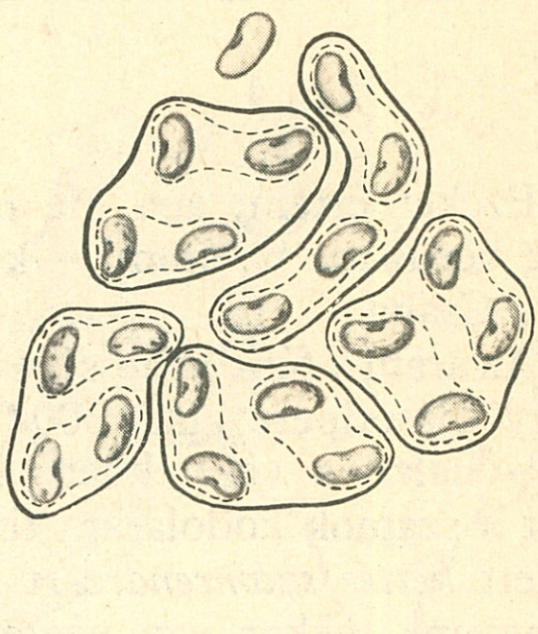


– Hadd számoljam meg én, Balázs! – kérlelte az öccsét Kinga, de Balázs hajthatatlan maradt.

– Kettesével körbe kell keríteni, mert kettes számrendszerben dolgozunk,

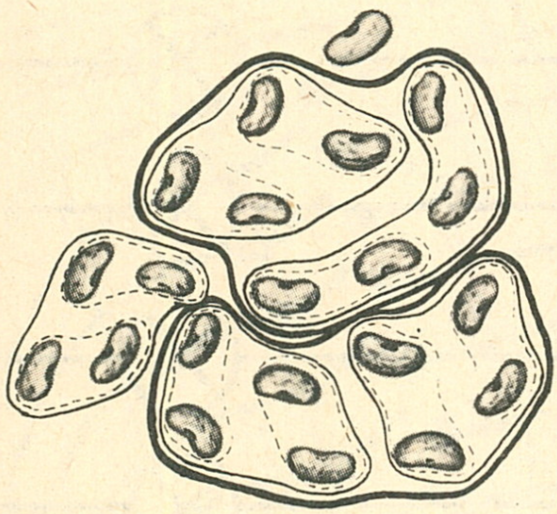
azután a kettes csoportokból kettesével újabb csoportokat készítek, ezek négyet érnek

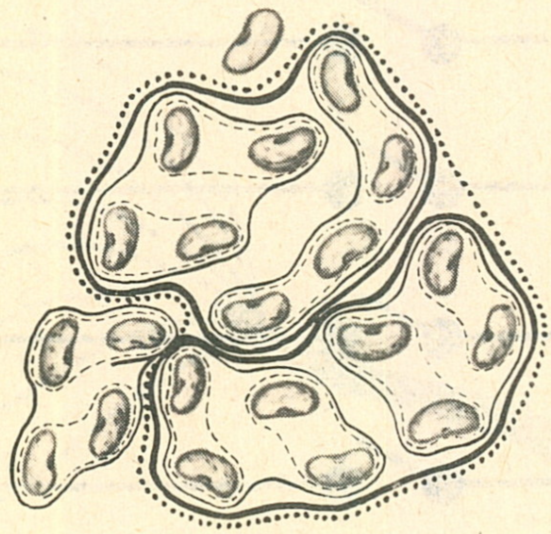




Két négyes csoportból lesz egy nyolcat érő csoport, ezeket ismét körbekerítem.

A két nyolcas csoportból is tudok készíteni még egy csoportot, az tizenhatot ér.





Most már felírom a számot:

16	8	4	2	1
1	0	1	0	1
1·16+	0·8+	1·4+	0·2+	1·1 = 21

– Helyes, Balázs, ügyes voltál! 10101 valóban egy kettes számrendszerbeli, röviden *bináris* szám.*

Figyelj ide, Csaba, most jön az áram. Minden egyes helyiértékhez képzelj el egy vezetéket.

* Gyerekek! A bináris számokat ezentúl ②-vel fogjuk jelölni. Pl.: 101②, 11② stb.

Jelentse az 1-et az, ha áram folyik át a vezetéken, és a 0-t az, ha nincs benne áram.

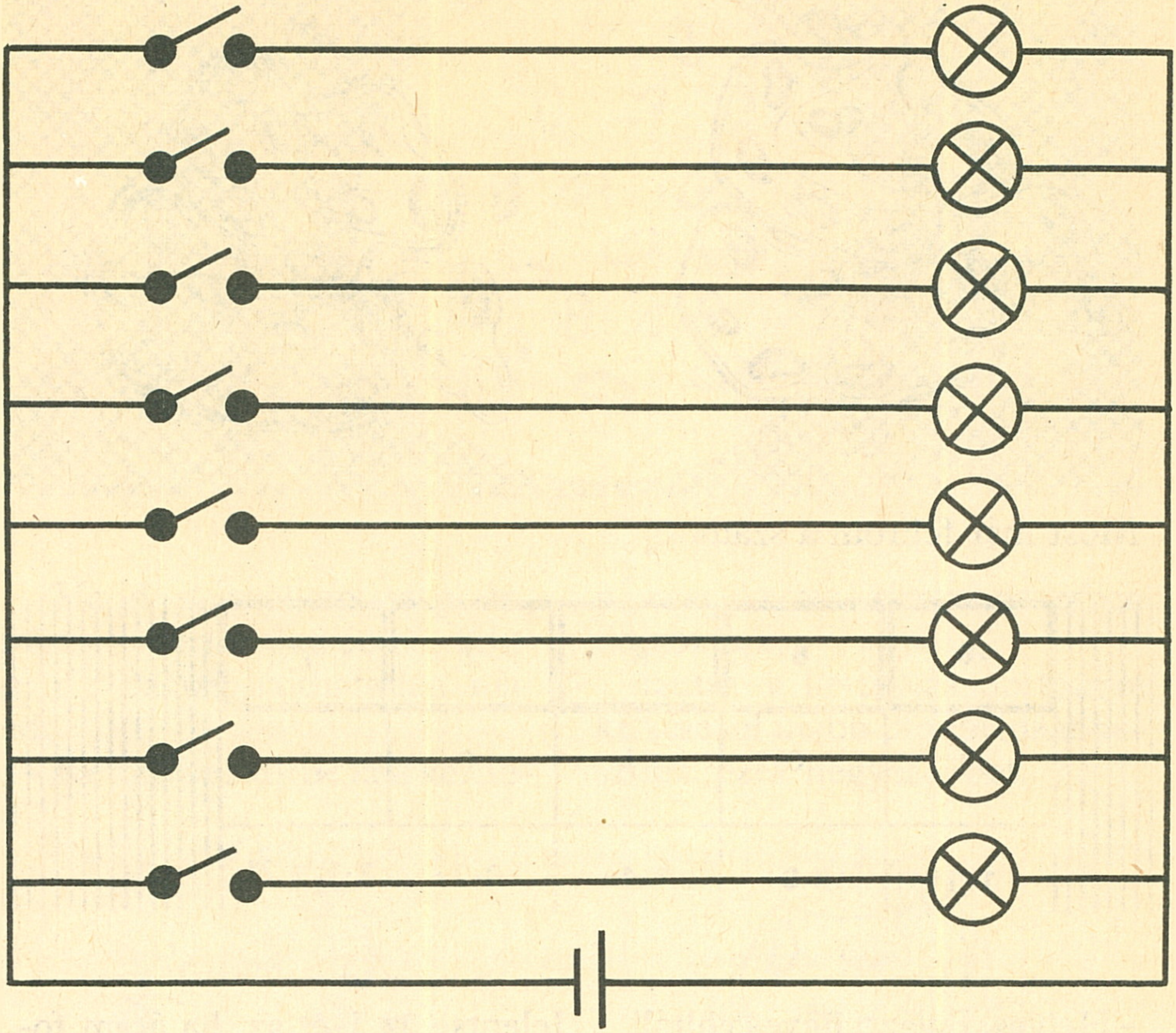
– Építsünk adatsínt! – javasolta Balázs, és ki tudja, honnan (Csaba szerint az ő versenyautójából), elővarázsolt egy elemet.

– Hány szál drót kell?

– Ahány helyiértékes számot akarsz továbbítani. De ha rám

hallgattok, csak nyolc szálat húztok ki. Szükség van még szálanként egy-egy, összesen nyolc darab kapcsolóra és nyolc darab izzó is kell.

Így kell összekötni őket:



A rajzon ez a jel —⁄ — jelenti a kapcsolót, ez ⊗ az izzólámpát és végül ez —|⊢— az elemet.

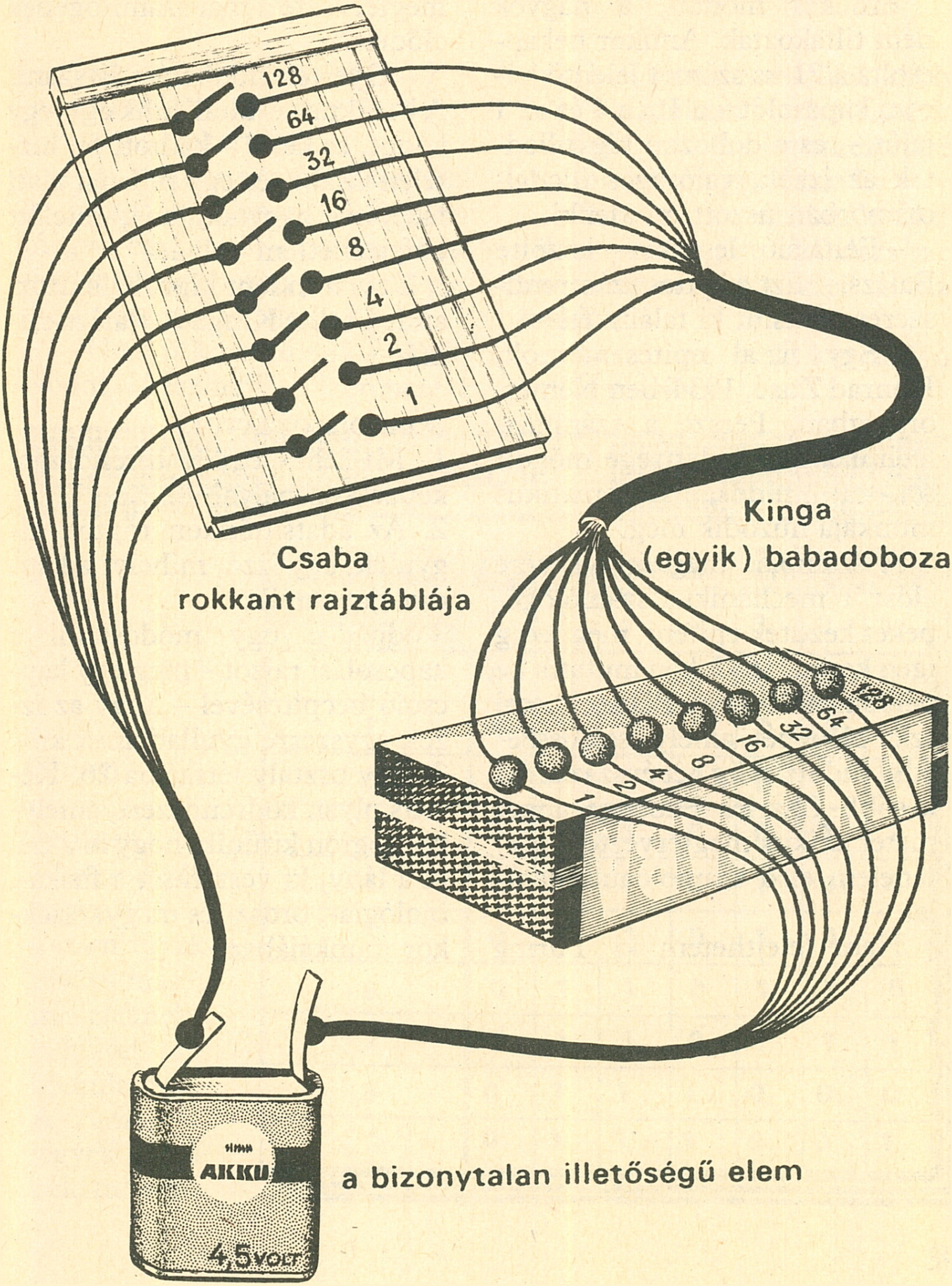
– Valahogyan meg kellene jelölni a helyiértékeket, mert nem fogjuk tudni jól leolvasni a számokat.

– Írjuk az izzók mellé a helyiértékeket: 1, 2, 4 és így tovább!

– Ez az!

– Csináljuk!

Nagyon nagy volt a lelkesedés. Némi huzavona után ez kerekedett ki a dologból:



– Én kapcsolom be elsőnek! – sietett kijelenteni Balázs.

Érdekes módon, a nagyok nem tiltakoztak. Amikor bekapcsolta a 21-es számot jelentő három kapcsolót – a 16, a 4 és az 1 jelűt –, és a dobozon kigyulladtak az izzók, valóságos diadalmámorban úszott mindenki.

– Feltaláló leszek – közölte Balázs. – Ezt a kettes számrendszeres adatsínt ki találta fel?

– Egy fiatal építészmérnök, Konrad Zuse, 1934-ben Németországban. Persze a számítástechnika mai fejlettsége mögött sok-sok tudós, matematikus munkája húzódik meg.

A számolás megkönnyítésére először mechanikus számológépeket kezdtek építeni, mégpedig igen korán. 1642-ben mutatta be például a 19 éves Blaise Pascal számológépét, amely csak összeadni tudott. Alig két évszázaddal később Charles Babbage angol matematikus megtervezte a történelem első számolóautomatáját.

Nem felejthetem ki Turing angol matematikus nevét sem, aki már századunk 30-as éveiben megtervezte a mai számítógépek elődeit.

– Gyertek titkosírást játszani! A betűknek is kitalálunk egy-egy kódot, és kész! – kiáltott fel hirtelen Csaba, és egy pillanat alatt faképnél hagytak, még mielőtt befejezhettem volna.

Ti is mindent kipróbálhattok, amit Csaba, Kinga és Balázs csinált!

GONDOLKOZZATOK!

1. Mi lehet egy autórendszám képzési szabálya?

2. Az adatsínünkön rögtön kigyullad egy izzó, mihelyt a kapcsolóját zárják.
Tudnád-e úgy módosítani a kapcsolási rajzot – még egy kapcsoló beépítésével – hogy az izzók egyszerre gyulladjanak ki?

3. Egy osztály létszáma 26. Készíts olyan kódrendszert, amelyből rögtön kitűnik, hogy ki a fiú, ki a lány, ki vesz részt a fizika-, biológia-, orosz- és magyarszakkör munkájában!

– Hogy sikerült a titkosírás? – érdeklődtem, mikor gyermekeim újból előkerültek.

– Á, nem ér semmit az egész! – fakadt ki Kinga. – Egyszer négy-, egyszer öt-, hol meg hatjegyű bináris szám jelentett egy betűt vagy más jelet!

– Nem is könnyű dolog egy jó kódot készíteni! Szerencsére nem kell kitalálnotok újat, többfélét használnak a számítástechnikában is. A legismertebbet úgy hívják, hogy „eszki”.

– Biztosan az eszkimók találták ki – ragyogott fel Balázs szeme, mert nagyon tiszteli az eszkimókat.

– Nem, ezt csak kimondani kell így, leírva

ASC II

Ebben minden betűnek – kicsinek és nagynak –, a számjegyeknek és más írásjeleknek is külön kódjuk van. A számjegyek kódja például ez:

számjegyek	helyiértékek száma						
	7	6	5	4	3	2	1
0	0	1	1	0	0	0	0
1	0	1	1	0	0	0	1
2	0	1	1	0	0	1	0
3	0	1	1	0	0	1	1
4	0	1	1	0	1	0	0
5	0	1	1	0	1	0	1
6	0	1	1	0	1	1	0
7	0	1	1	0	1	1	1
8	0	1	1	1	0	0	0
9	0	1	1	1	0	0	1

– Te ezt mind kívülről tudod, apu?

– A számjegyek kódját? Te is megjegyezheted: az ötödik, hatodik helyiérték – vagy bit – mindig 1, az első négy biten pedig maga a szám áll, bináris formában.

– Kipróbálom! – hitetlenkedett Kinga. Legyen a szám mondjuk 24, mert július 24-én lesz a névnapom.

16	8	4	2	1
1	1	0	0	0

24 binárisan
az 11000, és... Mit is kell most csinálnom?

– Áthúzod az egészet, édes lányom, mert minden írásjelnek külön kódja van, így külön kell kódolni a 2-est és utána a 4-est. Más szóval, kódolni *karakter*enként, azaz jelenként kell.

– Na jól van, akkor újból kezdem.

a 24 első karaktere 2.
2 binárisan **$10_{\textcircled{2}}$**,
ugyanez négy biten **$0010_{\textcircled{2}}$**.
Az ötödik, hatodik helyiérték csupa 1,

110010,

és végül a hetedik bit 0.

2 → **0110010.**

– Eltaláltam? Eltaláltam! Akkor a 4 kódja is **011**-gyel kezdődik, és a vége pedig **0100.**

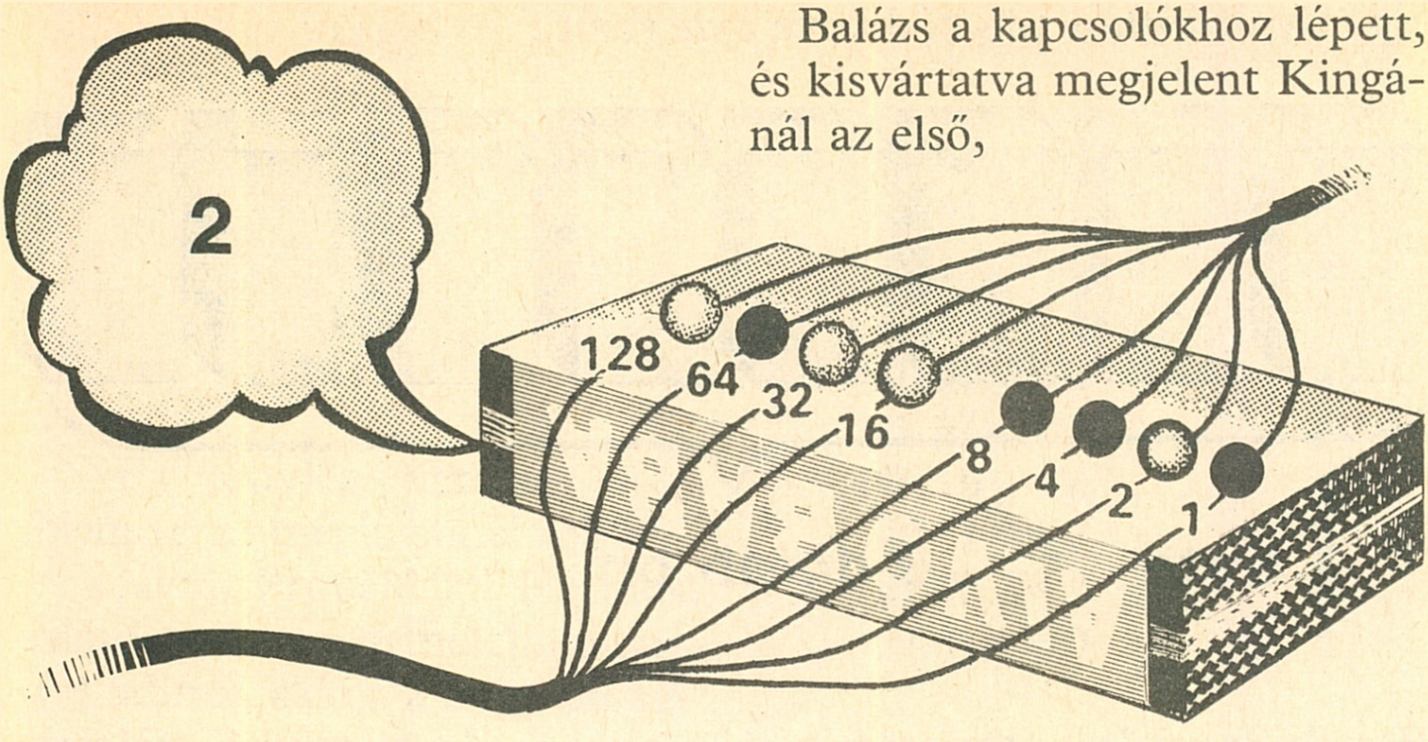
4 → **0110100.**

– Mire való az adatsín nyolcadik vezetéke, ha itt minden kód hétbites? – kérdezte Csaba.

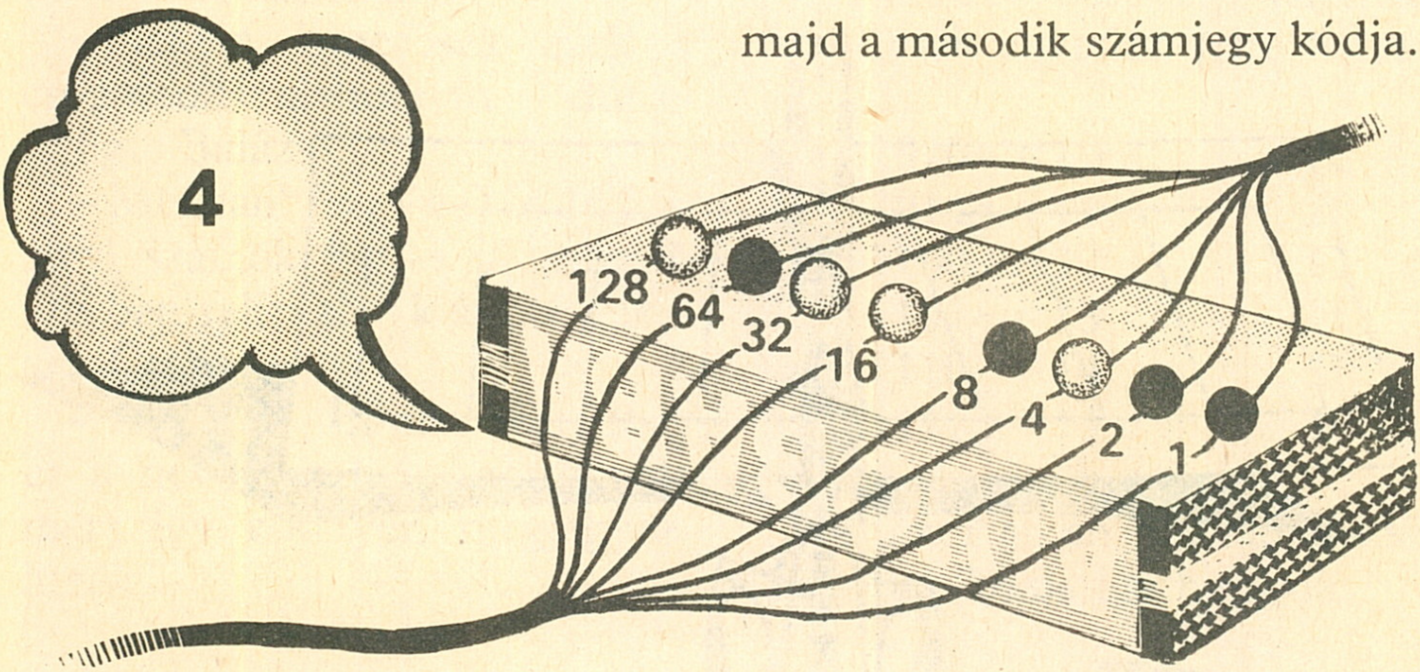
– A nyolcadik bit *adatvédelem*re szolgál. Könnyebben észre lehet venni az adattovábbítás közbeni hibákat, ha egy-egy kód mindig csak páros vagy páratlan számú 1-est tartalmaz. Egy számítógépen belül persze ez állandó: vagy mindig páros, vagy mindig páratlan.

– Páros 1-eseket kérek! – rendelkezett máris Kinga. – Az én névnapomról van szó!

Balázs a kapcsolókhoz lépett, és kisvártatva megjelent Kingánál az első,



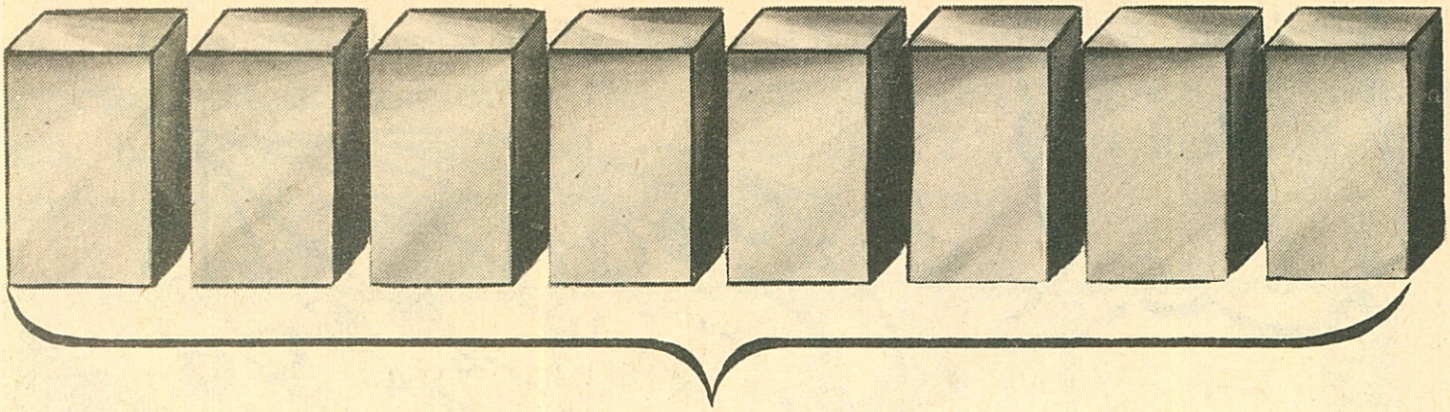
majd a második számjegy kódja.



– Hiába írtam fel olyan szépen a csoportok értékeit – szomorodott el Balázs –, most nem is érnek semmit, és kimondani is alig lehet ezeket a kódokat.

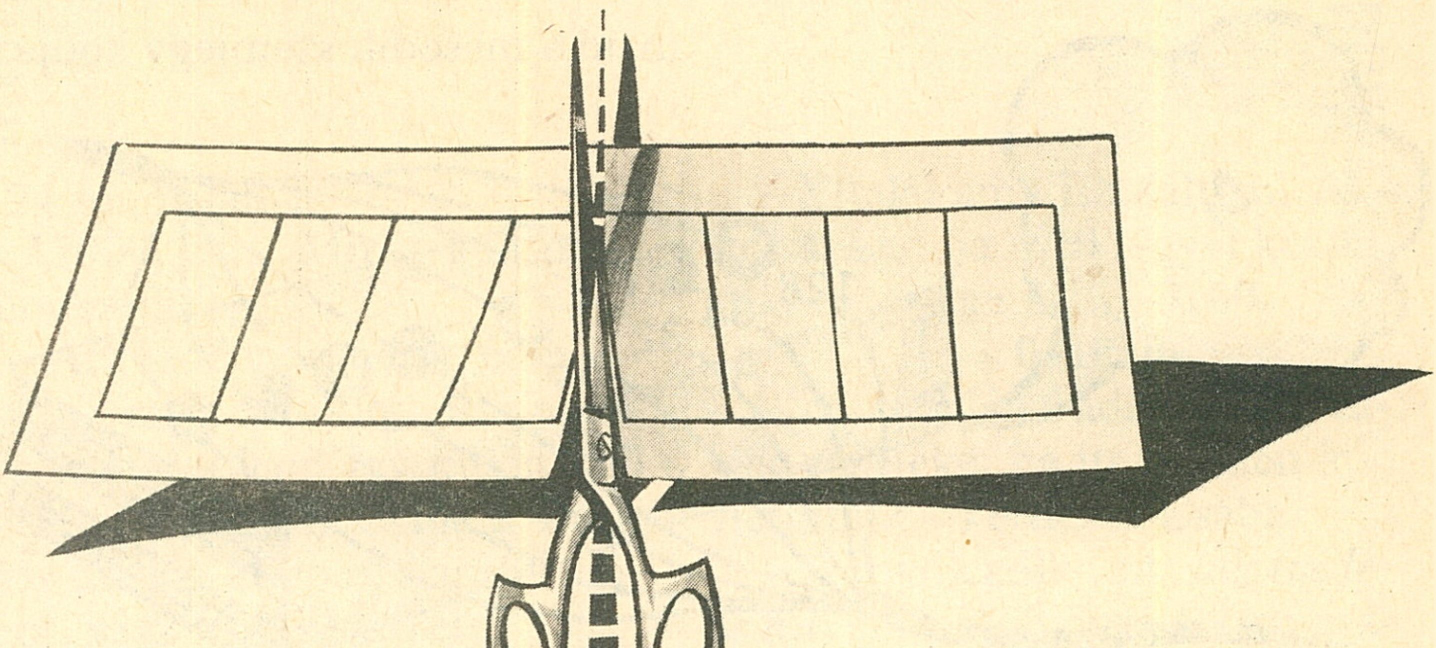
– Az igaz, hogy csak az 1, 2, 4 és 8-as csoportokat használják egy számjegy értékének leolvasásához, de a kódot kimondani elég könnyen lehet.

Egy nyolc bitből álló kód neve *byte* (bájt).



1 byte = 8 darab bit

Egy ilyen nyolcas csoport két négyes csoportra bontható:



Négy biten pedig a 0-ák és az 1-esek csak 16-féleképpen fordulhatnak elő:

0000
0001
0010
0011
0100
0101
0110
0111
1000

1001
1010
1011
1100
1101
1110
1111

– Ez tényleg tizenhat eset. Nem lehet mégis ennél több?

– Nem. Próbáld ki! Ha most mindegyik csoportnak külön nevet adunk, akkor a négy bit felsorolása helyett mindig csak egy nevet kell kimondani.

– Nekem ismerősek ezek a csoportok valahonnan – dünynyögte Balázs.

– Megvan! Ezek a 0 és 16 közötti számok kettes számrendszerben felírva! – kiáltott lelkesen Csaba.

– Úgy van. Éppen ezért 9-ig ez a nevük is a csoportoknak, mert így könnyen megjegyezhetők.

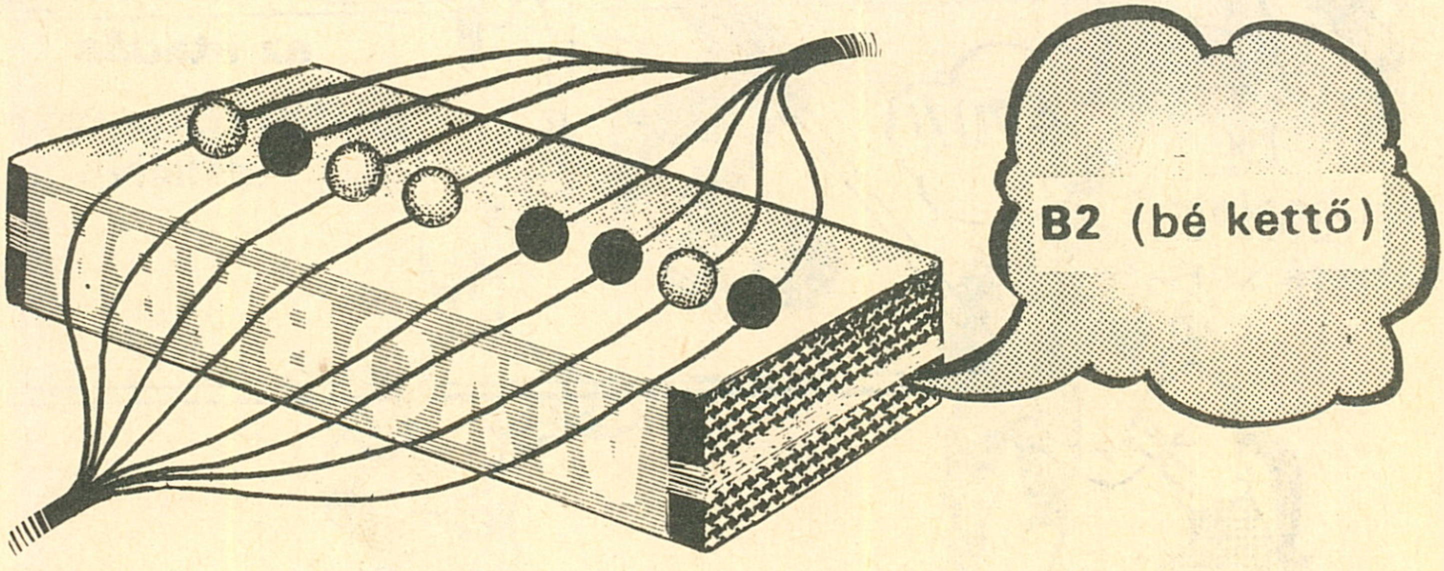
Az eset	A neve
$0000_{(2)} = 0$	0
$0001_{(2)} = 1$	1
$0010_{(2)} = 2$	2
$0011_{(2)} = 3$	3
$0100_{(2)} = 4$	4
$0101_{(2)} = 5$	5
$0110_{(2)} = 6$	6
$0111_{(2)} = 7$	7
$1000_{(2)} = 8$	8
$1001_{(2)} = 9$	9

– És a többivel mi lesz?

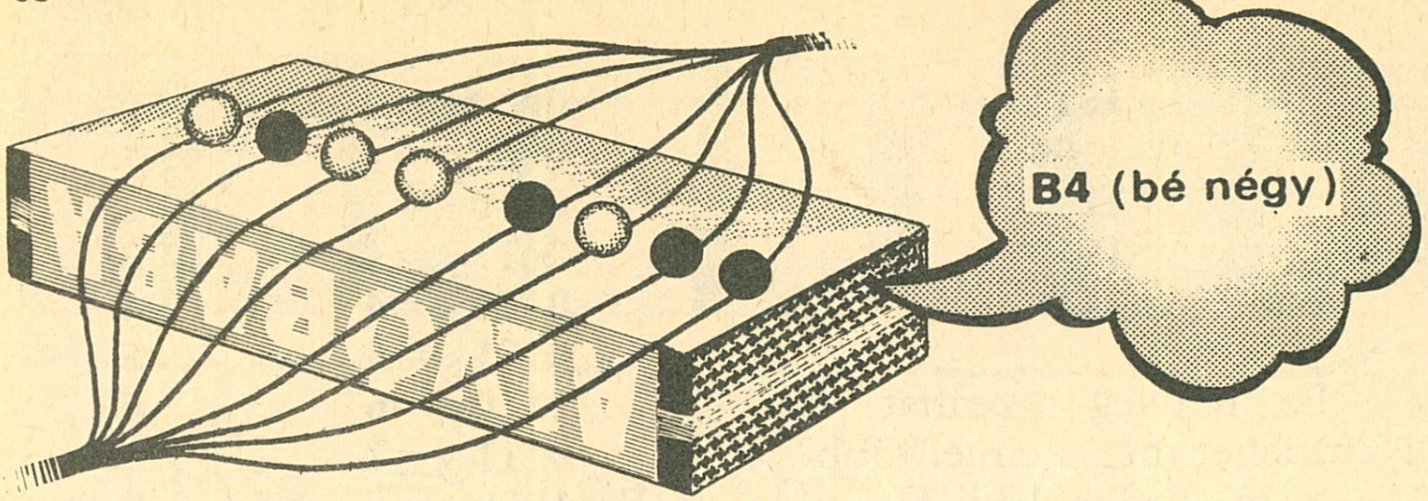
– Azokat betűvel jelöljük, hogy csak egyetlen jelre legyen szükség.

$1010_{(2)} = 10$	A
$1011_{(2)} = 11$	B
$1100_{(2)} = 12$	C
$1101_{(2)} = 13$	D
$1110_{(2)} = 14$	E
$1111_{(2)} = 15$	F

– Akkor az én névnapom dátumának kódja így hangzik:



és

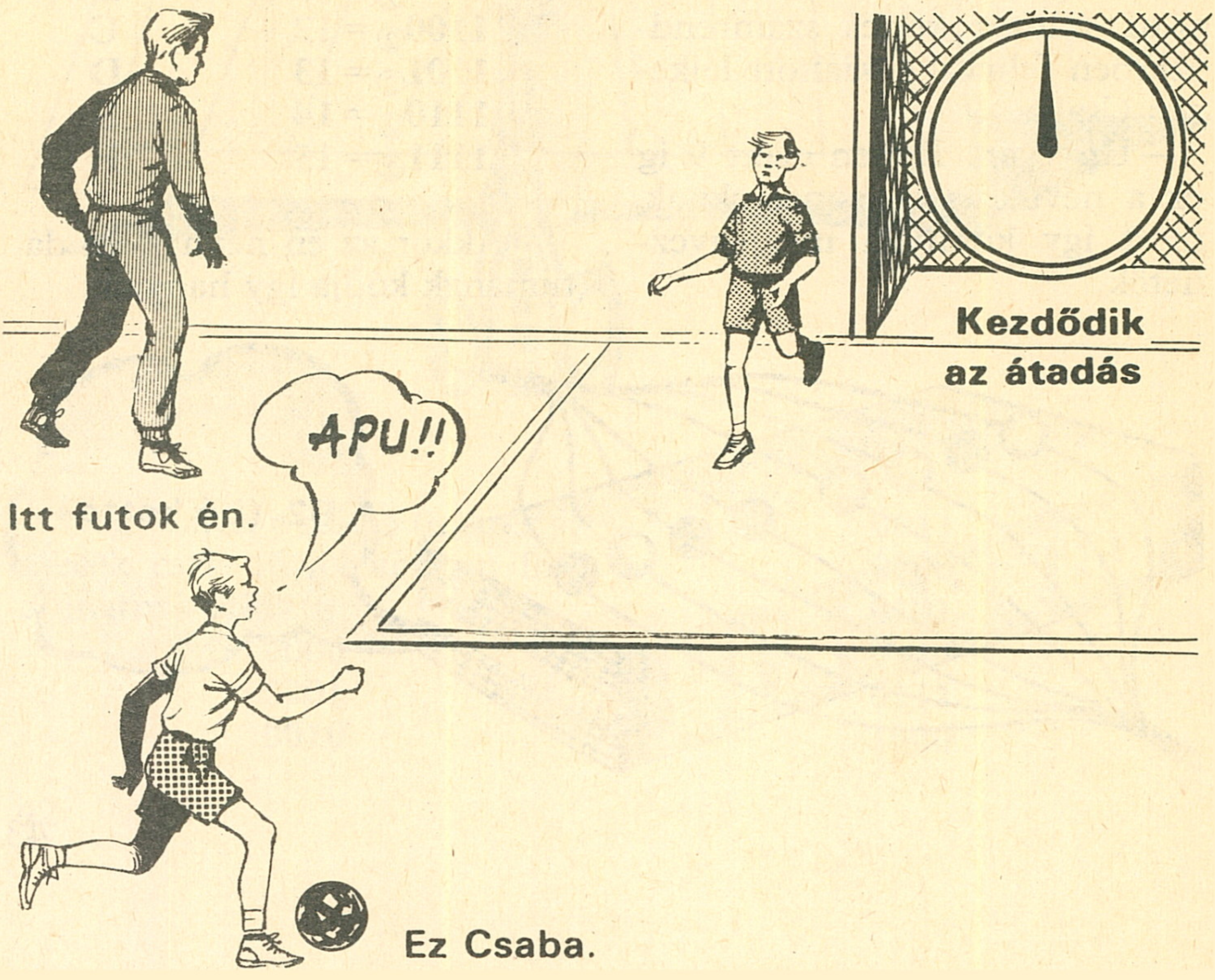


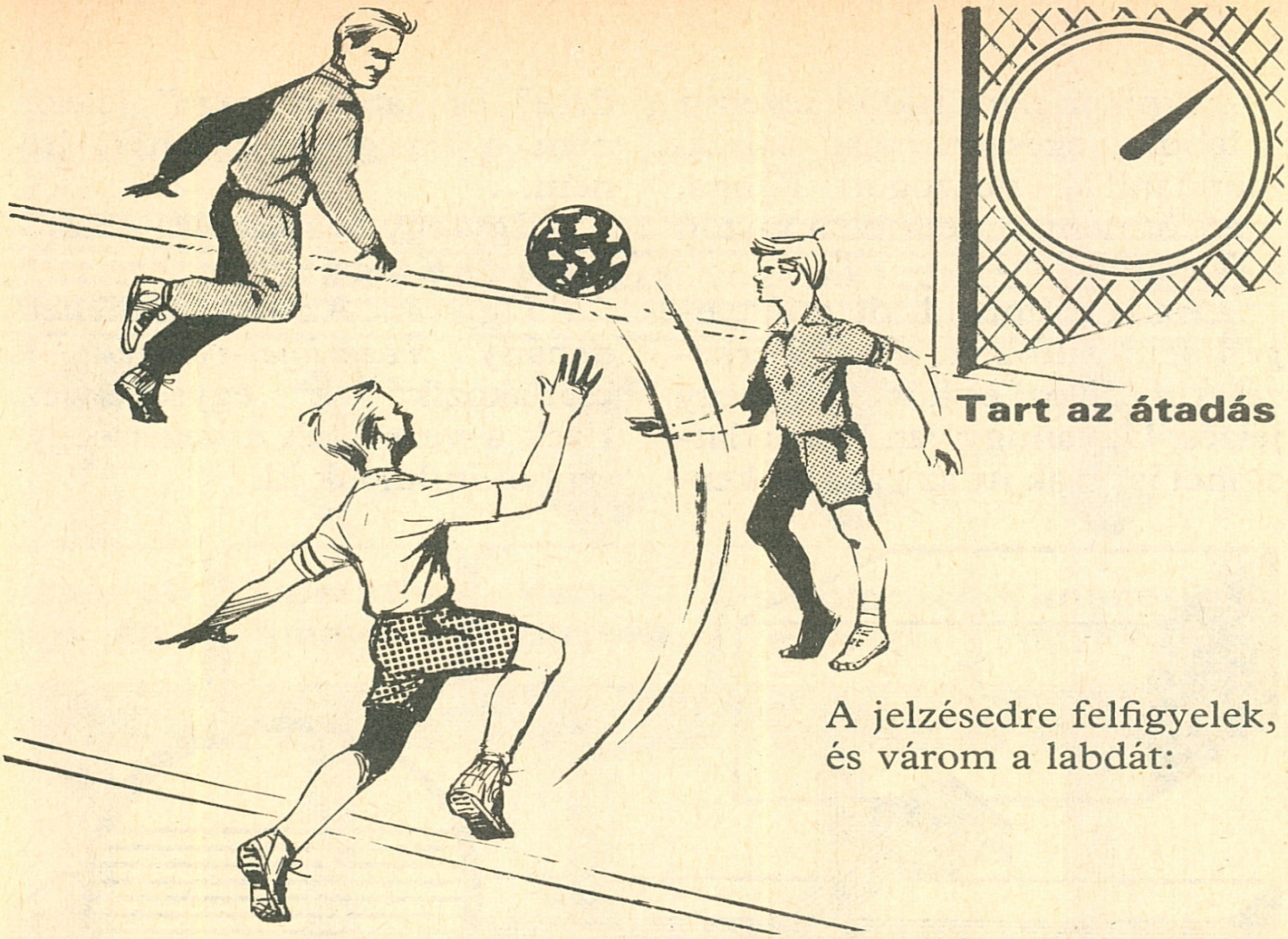
Van doboz, vagy nincs doboz?

– Apa, most megépítjük a dobozokat az adatoknak? – kérdezte a barkácsolók gyöngye.

– Attól tartok, az bajos lenne.

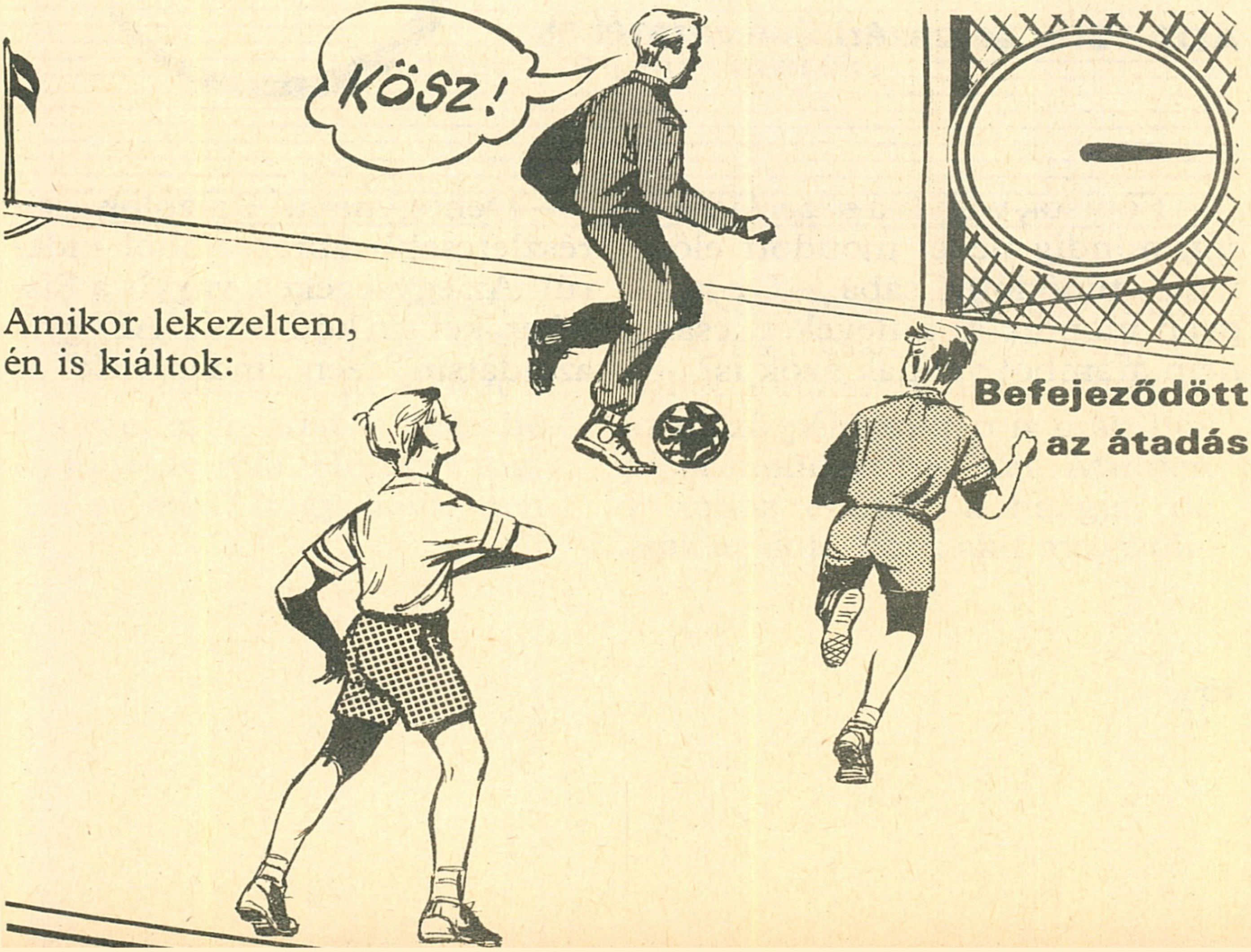
A doboz ugyanis nem más, mint az idő. Nem kell azonban valami bonyolult dologra gondolnod. Focizás közben, például, mielőtt passzolsz, többnyire kiáltani szoktál:





A jelzésedre felfigyelek,
és várom a labdát:

Amikor lekezeltem,
én is kiáltok:

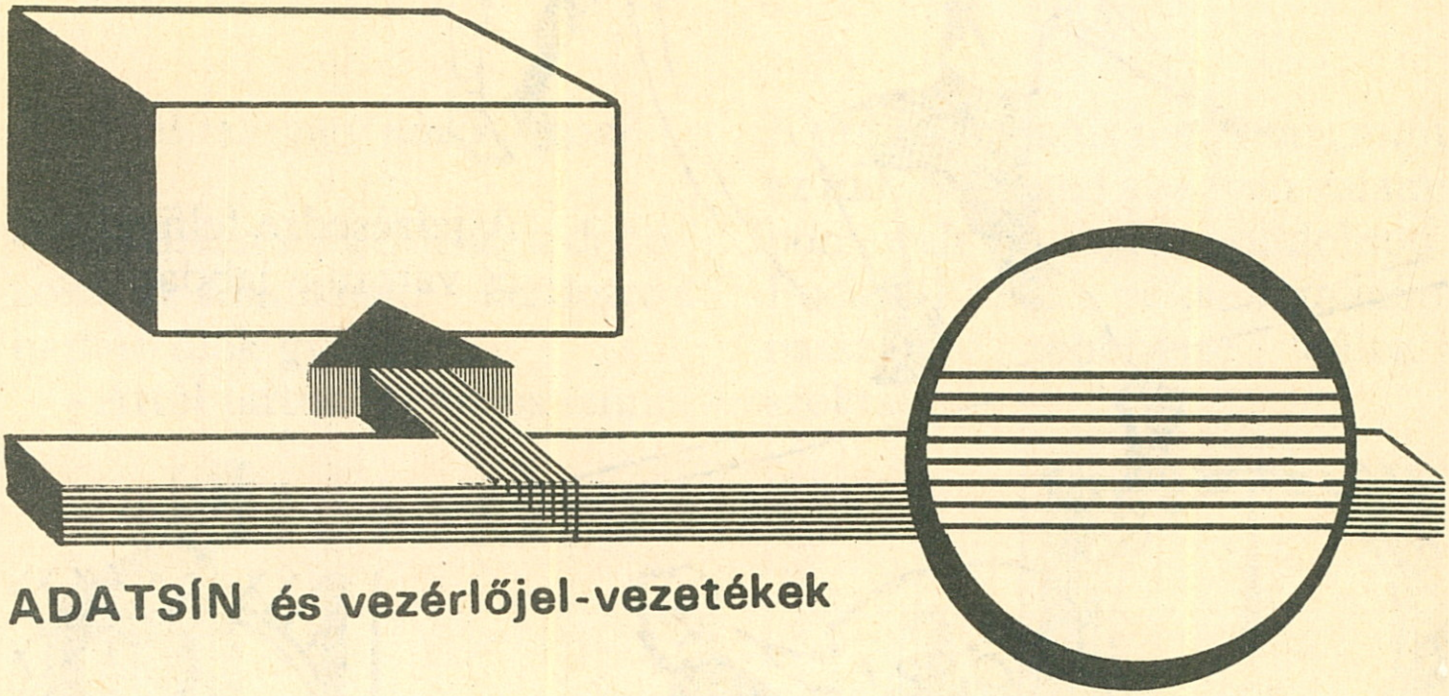


– Amikor nem tudod levenni a labdát, egészen mást szoktál mondani! – duzzogott Kinga, mert őt nem vettem be a rajzon a csapatba.

– Az is előfordul, de a lényegen nem változtat: a labda érkezését megelőzi és követi egy-egy jelzés. Ugyanúgy van ez az adatsínnél is, csak itt az „átadás kezdete" és „átadás vége" jeleket nem a levegő továbbítja, hanem...

– Egy-egy vezeték árammal – vágta rá Kinga.

– Úgy van. Az adatsín mellett néhány *vezérlőjel-vezeték* is csatlakozik az egységekhez. Ezek a vezetékek a vezérlőegységből indulnak ki.

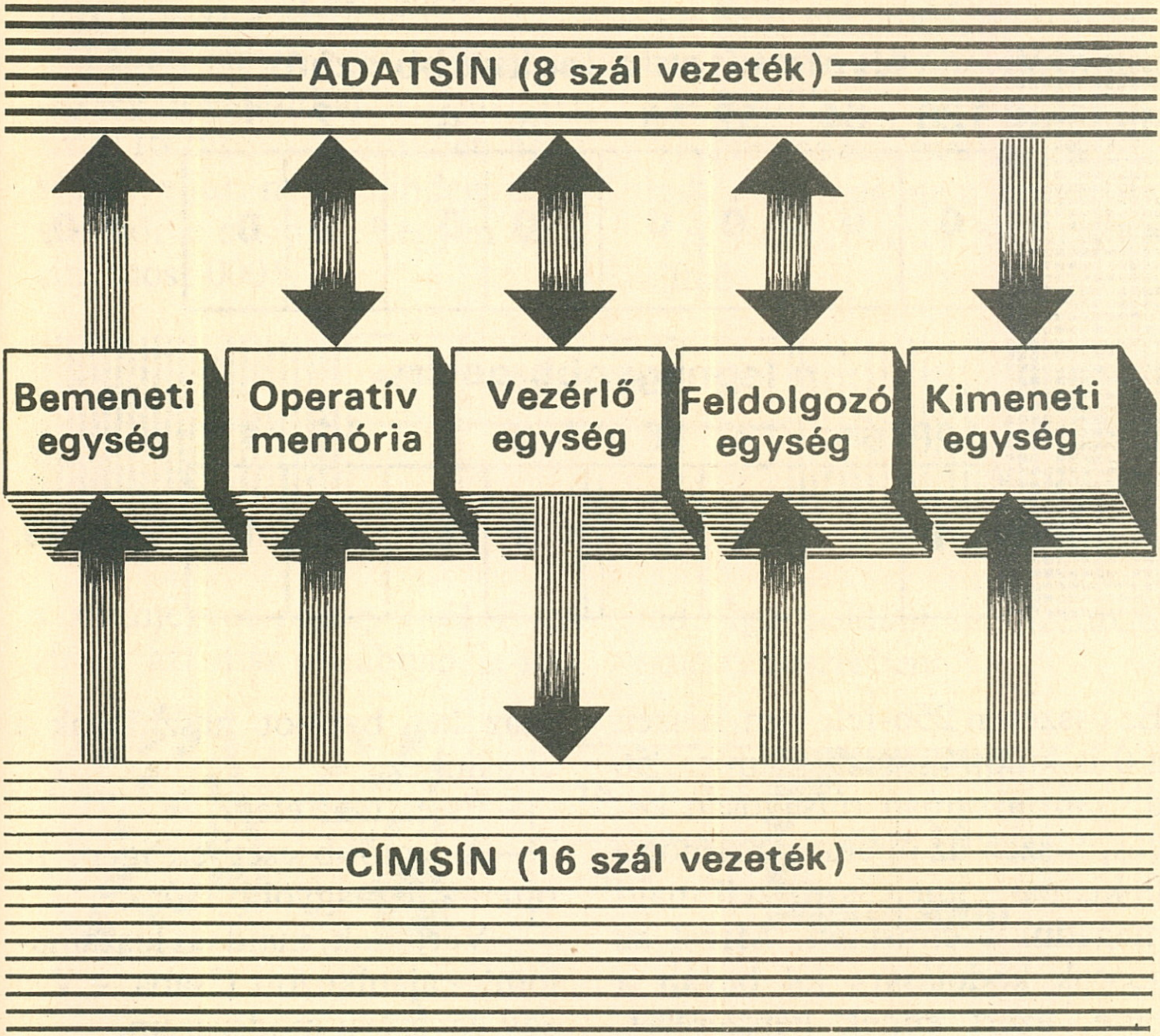


ADATSÍN és vezérlőjel-vezetékek

– Főpista, vagyis a *vezérlőegység,* mindig nevet mondott először – tűnődött Csaba. – De hogyan mondja ki a neveket, csak nem áramból vannak azok is?

– Dehogynem! Rajzolok egy részletesebb ábrát, abból kiderül. Az egységeket, vagyis a Pistákat, két sín köti össze. Az egyik az adatsín, ezen „mozognak" az

adatok, a másik a *címsín,* ezen pedig a *cím*ek, vagyis az egységek nevei. (a nyilak az áramlás irányát jelzik)



– Nem értem ezt az egészet! – türelmetlenkedett Kinga. – Négyen is küldhetnek itt adatot az adatsínre, és az adat is négy helyen jöhet le róla. Honnan fogja tudni az adat, hogy hová menjen?

– Nézd meg a rajz alsó részét is! Címet csak a vezérlőegység adhat a címsínre: mint ahogyan az előző példánkban is csak Főpista mondott neveket. Minden Pistához eljutott a hangja, de csak az dolgozott, akit megszólí-

tott. Az egységek címei – vagyis nevei – 16 bites bináris számok.

– Miért éppen 16 bites?

– És miért nem elég a 8 bit, úgy, mint az adatsínnél?

– Nézzetek ide.

Nyolc biten a legkisebb szám

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	= 0,

a legnagyobb pedig

128	64	32	16	8	4	2	1	
1	1	1	1	1	1	1	1	= 255.

Ez összesen 256-féle cím, hiszen a 0 is külön címnek számít. 256 db cím azonban túlságosan kevés, hiszen az operatív memória fiókjainak címeit is itt kell majd megadni a címsínen. Mivel az adatok kódolására jól bevált a byte, ezért ennek kétszeresét, azaz két byte-ot használnak a címsínhez.

– És ez már elég?

– A legtöbb esetben igen. 16 biten a legnagyobb szám…

– Várj csak, majd én kiszámolom! – ajánlkozott Csaba. – Van egy egyszerű módszerem.

A 256 az éppen 2 nyolcadik hatványa.

$$2^8 = 256.$$

Most 8 bit helyett 16 bitet használunk, ezért

$$2^{16} = 65\,536.$$

– Ezt ügyesen csináltad! Most pedig térjünk vissza a címeinkhez, mert Kinga még nem kapott választ a kérdésére.

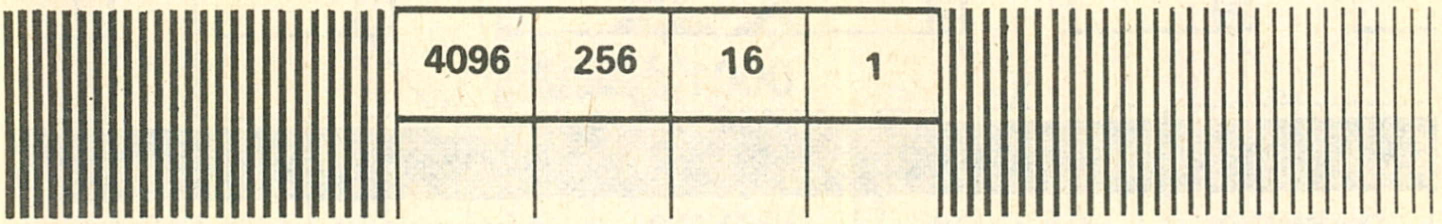
Tehát: címet csak a vezérlőegység továbbíthat a címsínre. Ez a 16 bites kód minden egységhez eljut, mert mindegyik rá van kötve a sínre. Legyen ez a cím most 0001.

– Ezt milyen számrendszerben kell érteni? Tízesben vagy kettesben? – kérdezte Kinga.

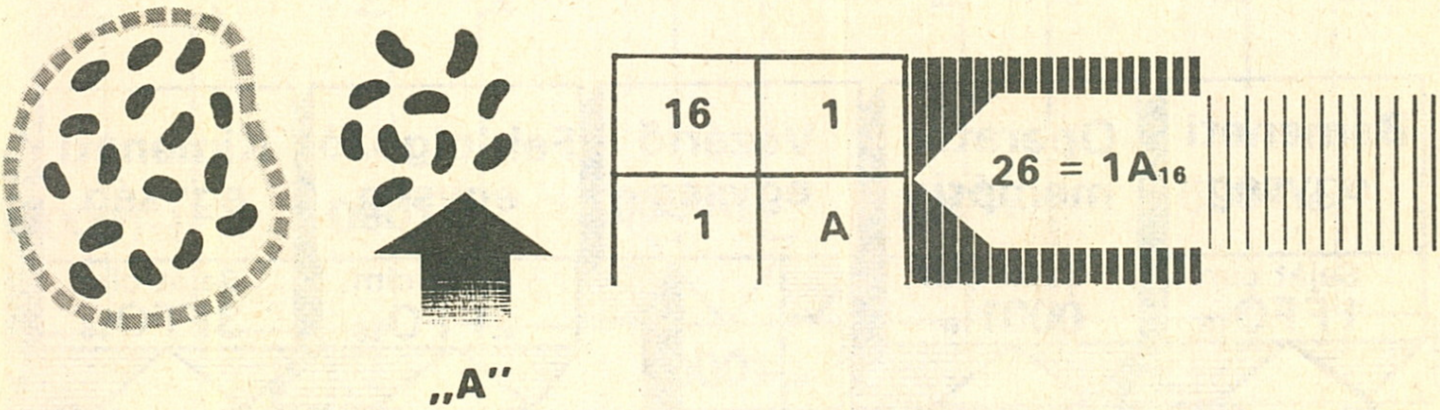
– Sem egyikben, sem másikban, hanem tizenhatosban.

– Apu!!!

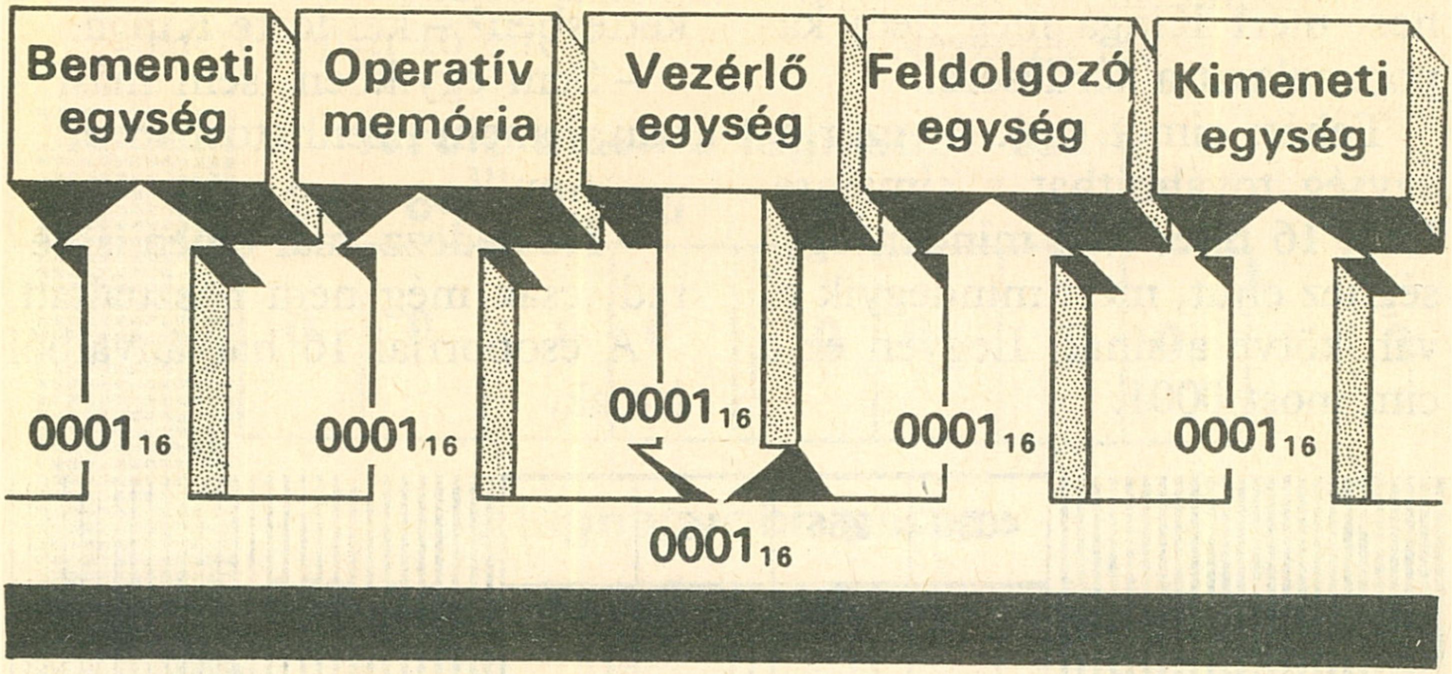
– Ne ijedezz, már régen ismered, csak még nem használtad. A csoportjai 16 hatványaiból állnak,



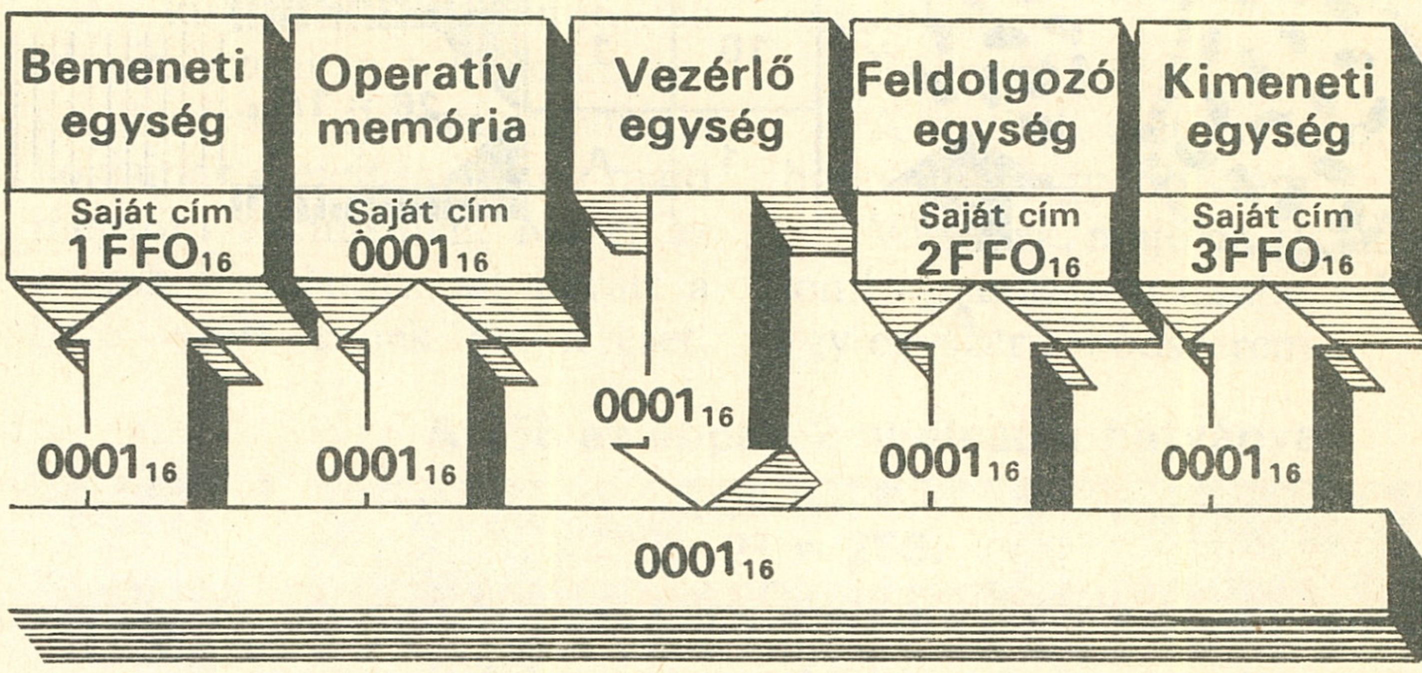
a számjegyei pedig 0-tól F-ig, ahogy azt már korábban is láttad. Így például 26 tizenhatos számrendszerben:



– Értitek? Legyen tehát a cím 0001(16)



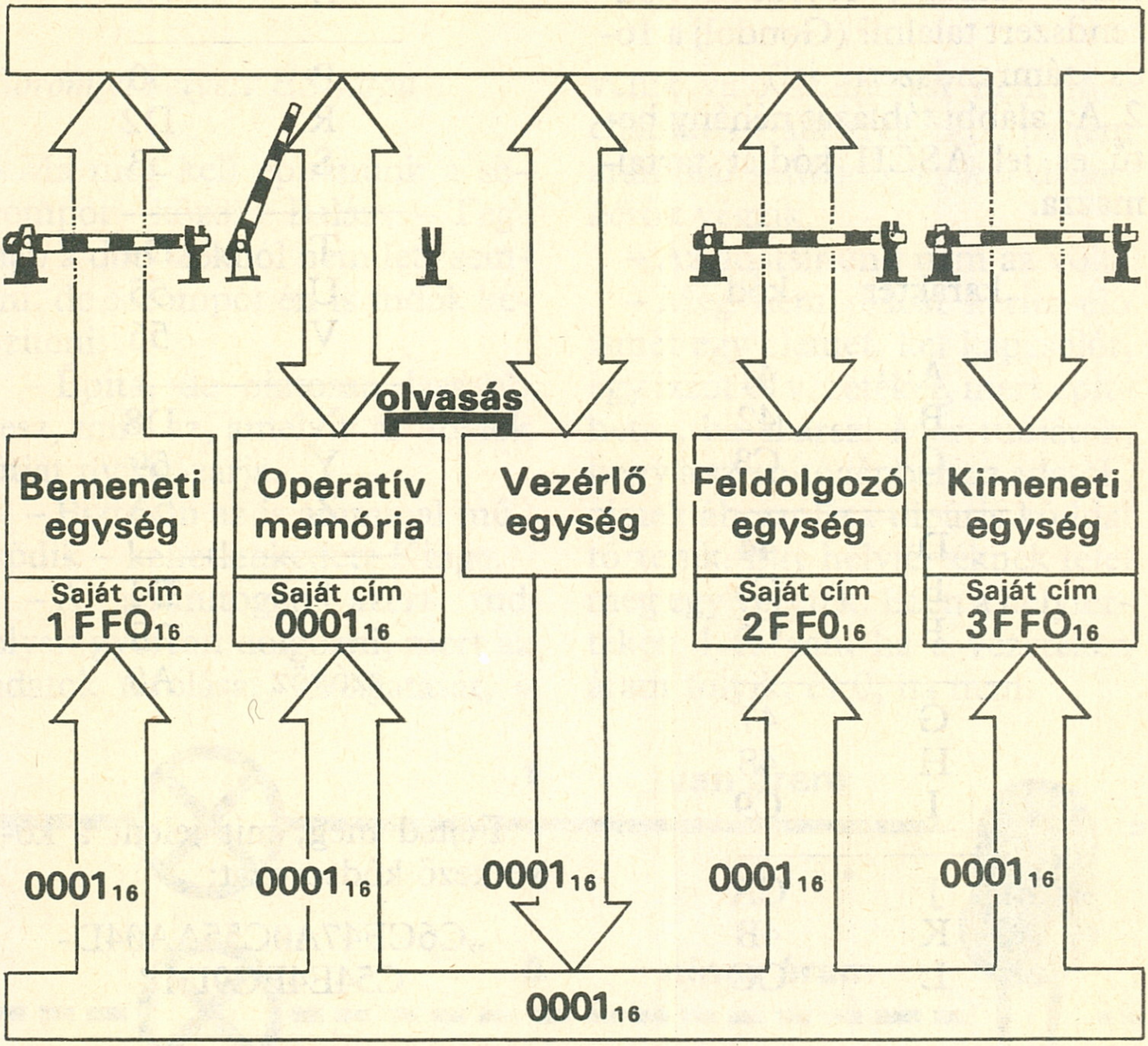
A címsín az egységekhez egy címfelismerő *logikai áramkör*ön keresztül csatlakozik, amely összehasonlítja az érkezett címet a saját címmel, és egyezés esetén az egység dolgozni kezd.



– Akkor most a memória fog dolgozni! – örvendezett Balázs.

– Igen. Az adatok útját pedig minden egység maga vezérli: ha

felismerte a címet, akkor kinyitja a „sorompót”, vagyis az adatutat, egyébként pedig nem.



– A rajzon láthattok egy vezérlőjel-vezetéket is, *olvasás* van ráírva. Nem ismerős nektek az, ami itt a rajzon történik?

– Ez ugyanaz, mint amikor a Főpista elkezdte a munkát! „Pista I, 1-es ki!” – vezényelte.

– Tényleg! Az 1-es cím jelenti az operatív memóriában az első fiókot, az olvasás jel meg azt, hogy ki kell venni, ami benne van!

– Ha kivette, mi marad benne?

– És mi az a logikai... amiről az előbb beszéltél?

– Megtudjátok a következő fejezetből!

GONDOLKOZZATOK!

1. Az előző fejezet utolsó példájához tudnál-e rövidebb kódrendszert találni? (Gondolj a 16-os számrendszerre is!)

2. Az alábbi táblázat néhány betű és jel ASCII kódját tartalmazza.

karakter	kód
A	41
B	42
C	C3
D	44
E	C5
F	C6
G	47
H	48
I	C9
J	CA
K	4B
L	CC
M	4D
N	4E
O	CF
P	50
R	D2
S	53
T	D4
U	55
V	56
X	D8
Y	59
Z	5A
!	D1
.	2E
szóköz	AØ

Fejtsd meg, mit jelent a következő kódsorozat:

„C6CF47AØC55AAØ4D-C54E4EC9D1”

Sorompó helyett ÉS kapu

– Ma meg kell építenünk a sorompót – közölte Balázs. – Tegnap a dobozokból nem lett semmi, de sorompót én is tudok készíteni.

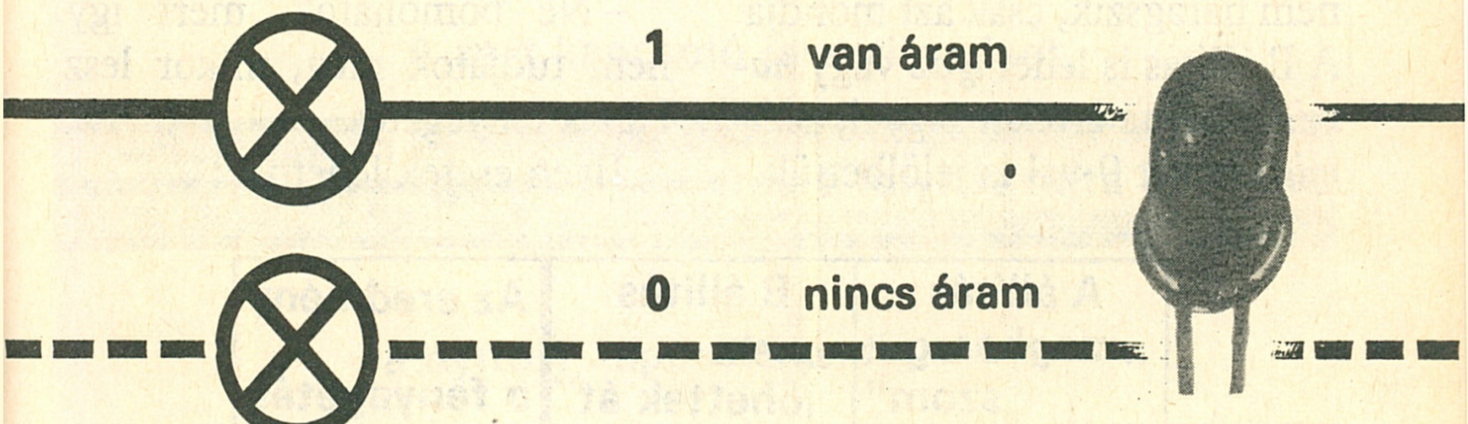
– Építs, de biztosan lassabb lesz, mint az, amelyik az adatok útját nyitja-zárja.

– Biztosan az is árammal működik – kelletlenkedett Kinga.

– A számítógép azért tud olyan gyorsan dolgozni, mert az adatok tárolását, mozgatását, a velük való számolást nem mechanikus alkatrészek, hanem árammal működő *logikai áramkörök* végzik.

– Az adatsínünk nem az volt?

– Még nem. Most keríts elő ismét egy elemet, két kapcsolót, egy izzót és vezetéket, mert építhetsz, ha akarsz! Már tudjátok, hogy a számítógépben az adatok, címek ábrázolása bináris kóddal történik. Egy helyiértéknek felel meg egy vezeték. Ezen a helyiértéken 1-es van, ha a vezetéken áram folyik, és 0, ha nem.



– Igen ám, de nem tudjuk, hogy mikor van ott 0, vagy mikor van ott 1-es.

– Nem is kell azt tudnod: olyan áramkörre van szükség, amelyik mindkét esetben jól működik. A matematika egyik területének, a *matematikai logikának* a szabályaival meg lehet tervezni egy ilyen áramkör működését, és aztán néhány fajta alapáramkörből mint építőkockából össze lehet rakni. A legegyszerűbb ilyen alapáramkör az

ÉS kapu. Azért ez a neve, mert ilyen típusú kapcsolatokat valósít meg:

„A állítás **ÉS** B állítás **ÉS**…" és így tovább.

– Megharagszom, és nem jöhettek át játszani a szobámba, ha nem adjátok vissza a ceruzámat! – szólt rá Kinga a fiúkra.

– Tessék, csak véletlenül maradt nálam – így Balázs.

– Várjunk csak, Kinga most egy ÉS kapcsolattal fenyegetett!

– Tényleg!

„megharagszom" ÉS „nem jöhettek át"

A állítás — **B állítás**

– Gondolkodjunk, mikor lesz ez az egész fenyegetés igaz? Az A állítás lehet *igaz* (mert tényleg haragszik), de lehet *hamis* is: nem haragszik, csak azt mondja. A B állítás is lehet *igaz* vagy *hamis*. Az *igaz* értéket 1-gyel, a *hamis* értéket 0-val is jelölhetjük.

– Ez csak hamis lehet, mert eddig még mindig beengedett minket játszani!

– Hát ezután nem foglak!

– Ne bomoljatok, mert így nem tudjátok meg, mikor lesz igaz a fenyegetés.

Ilyen esetek lehetnek:

A állítás „megharag-szom"	B állítás „nem jöhettek át"	Az eredmény a fenyegetés
hamis (0)	hamis (0)	hamis
hamis (0)	igaz (1)	hamis
igaz (1)	hamis (0)	hamis
igaz (1)	igaz (1)	igaz

– Láthatjátok: a fenyegetés akkor és csak akkor lesz igaz, ha mindkét állítás is igaz. Az ilyen táblázatot igazságtáblázatnak nevezik, és rövidebben így lehet felírni:

Bemenetek		Kimenet (jele: Q)
A	B	Q = A ÉS B
0	0	0
0	1	0
1	0	0
1	1	1

Most pedig elkészítjük azt az áramkört, amelyik pontosan így működik. A bemeneteknek megfelel egy-egy kapcsoló.

A zárt kapcsoló jelenti az 1-et,

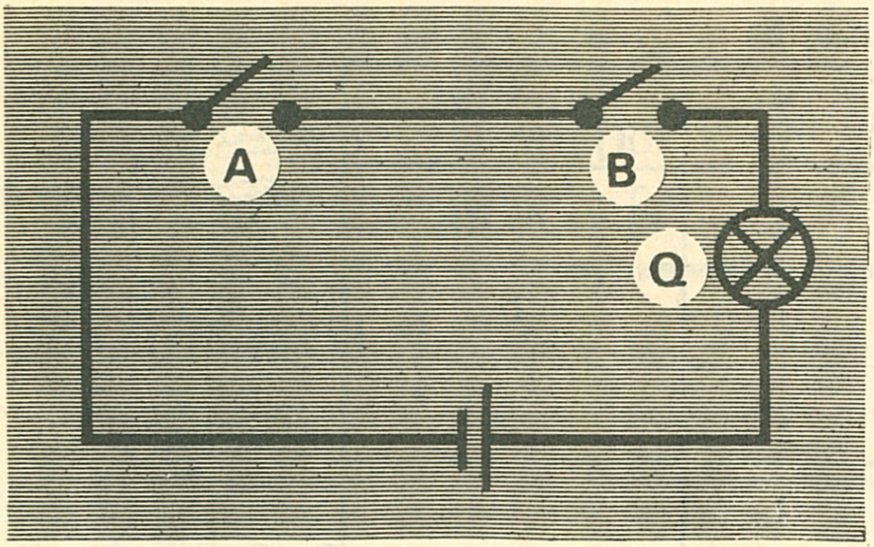


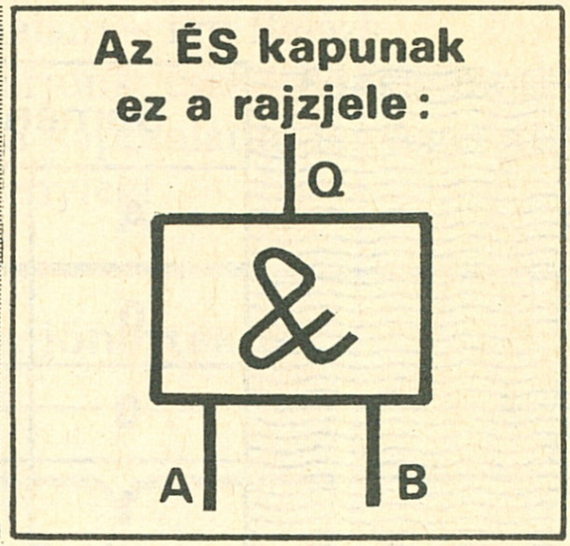
a nyitott kapcsoló pedig a 0-t,



– A kimenet pedig az izzó, éppen úgy, mint az adatsínnél.

– Igen. Így kell elkészíteni kapcsolókkal ezt az ÉS kaput:

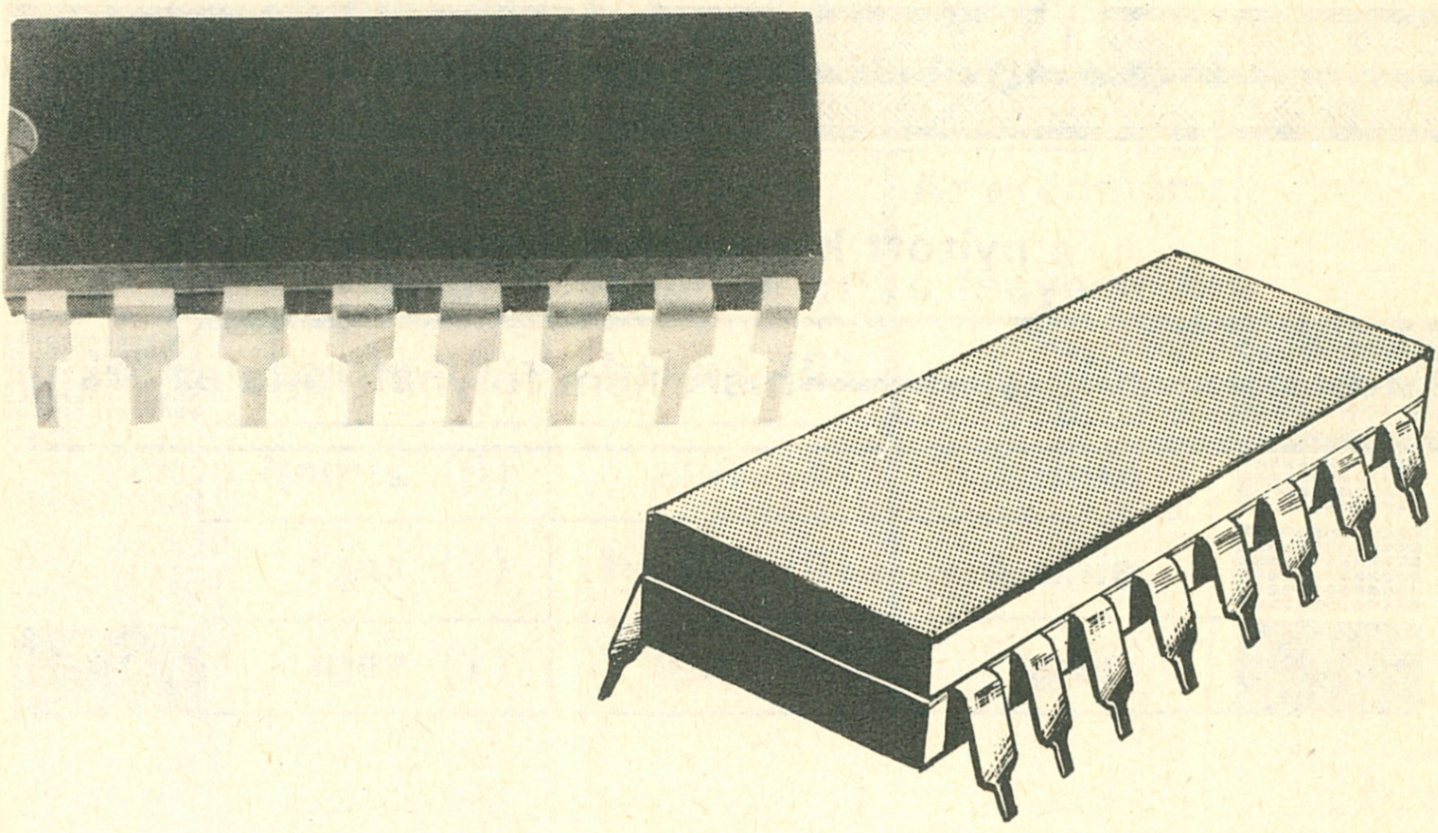




– Jaj de jó! Ilyenek vannak a számítógépekben is?

– Nem, dehogy. Azok diódákat, tranzisztorokat használnak kapcsolók helyett. A diódák és tranzisztorok sokasága szabványos méretű kis dobozokba van beépítve.

A dobozokból – pontos nevükön tokokból – kis fémlábak állnak ki, ezek a be-, illetve kimenetek.



Egy ilyen tok akkora, mint, mondjuk, egy gombfesték, de nem egyetlen áramkör van ám benne, hanem...

– Apu, mutass még ilyen ajtókat!...

– Sokféle van ám, Balázs, de azért egyet még megmutatok. De ne ajtónak hívd, hanem kapunak, sőt VAGY *kapu*nak. Szabálya:

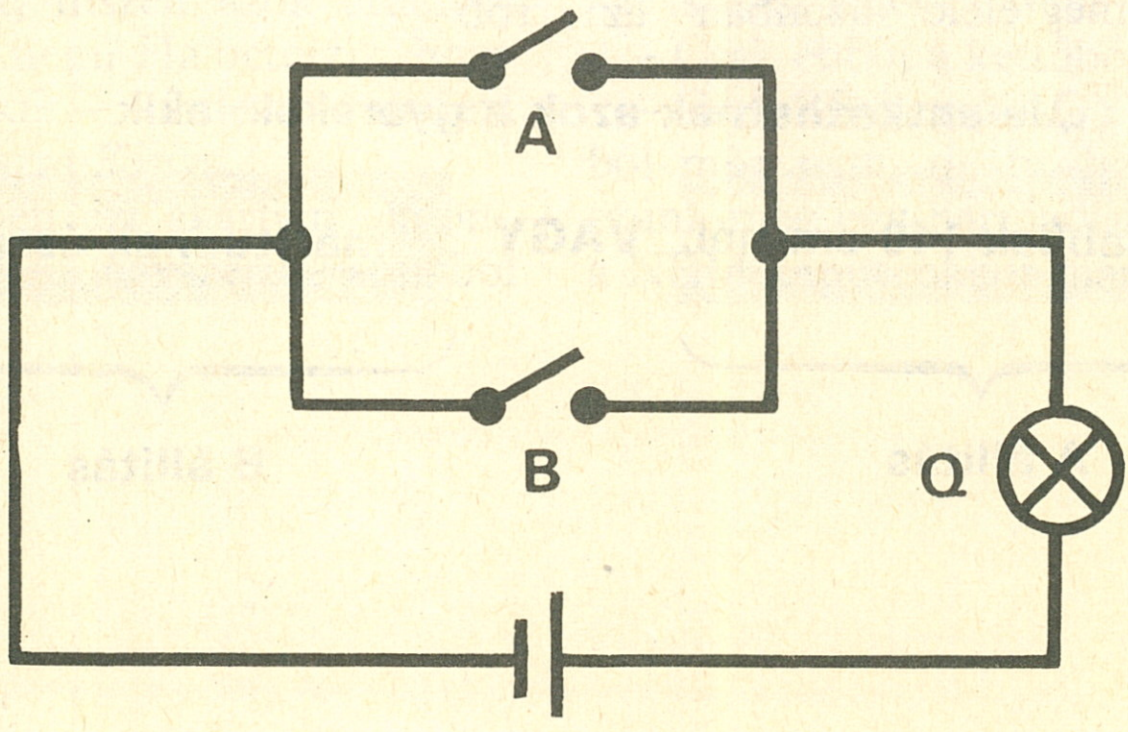
„A állítás **VAGY** B állítás **VAGY**...” és így tovább.

Most felírom egy kétbemenetű VAGY kapu igazságtáblázatát, ti pedig próbáljatok példát mondani rá!

Itt az igazságtáblázat:

Bemenetek		Kimenet
A	B	Q=A VAGY B
0	0	0
0	1	1
1	0	1
1	1	1

Ez a kapcsolás áramköre:



– Van-e már példa?

– Van egy, de nem biztos, hogy jó:

„Rabok legyünk, vagy szabadok?”

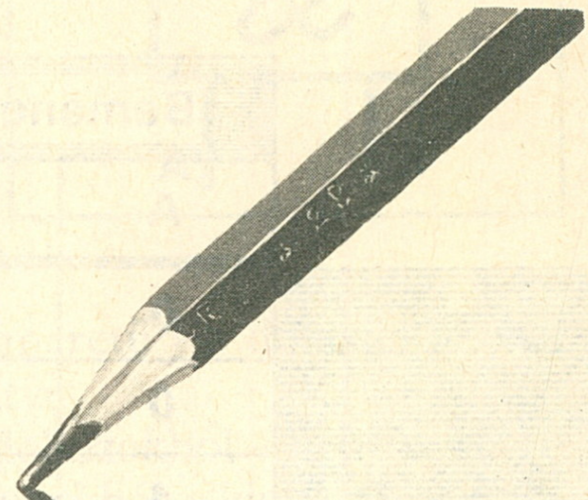
– Ez tényleg nem jó, de csak egy hajszálon múlott: itt nem lehet mindkét állítás egyszerre igaz. Ez kiderül egyébként a folytatásból is:

„Ez a kérdés, *válasszatok!*”

Azért ne kedvetlenedjetek el, ilyen kapu is van: a neve KIZÁRÓ VAGY.

Esetünkben azonban mindkét állítás egyszerre is igaz lehet.

Bemenetek		Kimenet
A	B	Q=A VAGY B
0	0	0
0	1	1
1	0	1
1	1	1



Ezért nem jó a példátok. Tökéletesen megfelelő azonban az úszótábor hirdetése a faliújságról:

„Jelentkezhetnek azok a gyerekek, akik

magasabbak 140 cm-nél, VAGY már tudnak úszni.”

A állítás **B állítás**

Ha belegondoltok, észrevesziteк, hogy az a gyerek is jelentkezhet, aki mindkét feltételnek megfelel: magasabb is, mint 140 cm, és úszni is tud.

– Azért választottunk rosszul, mert mi csak a „VAGY” szócskát kerestük!

– Elhiszem. Éppen itt van a nagy különbség az élő magyar nyelv és a matematikai logika között. A nyelv csodálatosan gazdag kifejezésekben, így csak a mondat értelme mutatja meg, hogy milyen logikai kapcsolattal van dolgunk, nem pedig a benne szereplő szavak. Ne feledkezzetek el azonban arról, hogy egy számítógép-áramkör működését leíró mondatban csak egyértelmű, világos kifejezések vannak. Például:

pl:

„Ha az adatsín 5. bitje 1 ÉS a 6. bitje 1 ÉS a 7. bitje 0, akkor a kód számot jelöl.”

– Apa! Te itt beszélsz, beszélsz az áramkörökről, én pedig azt szeretném tudni, hogy mik a fiókok, hogyan teszik bele az adatokat, hiszen azok áramból vannak, nem? Ha beteszik, hogy veszik ki? – kérdezte mindezt egy szuszra Kinga.

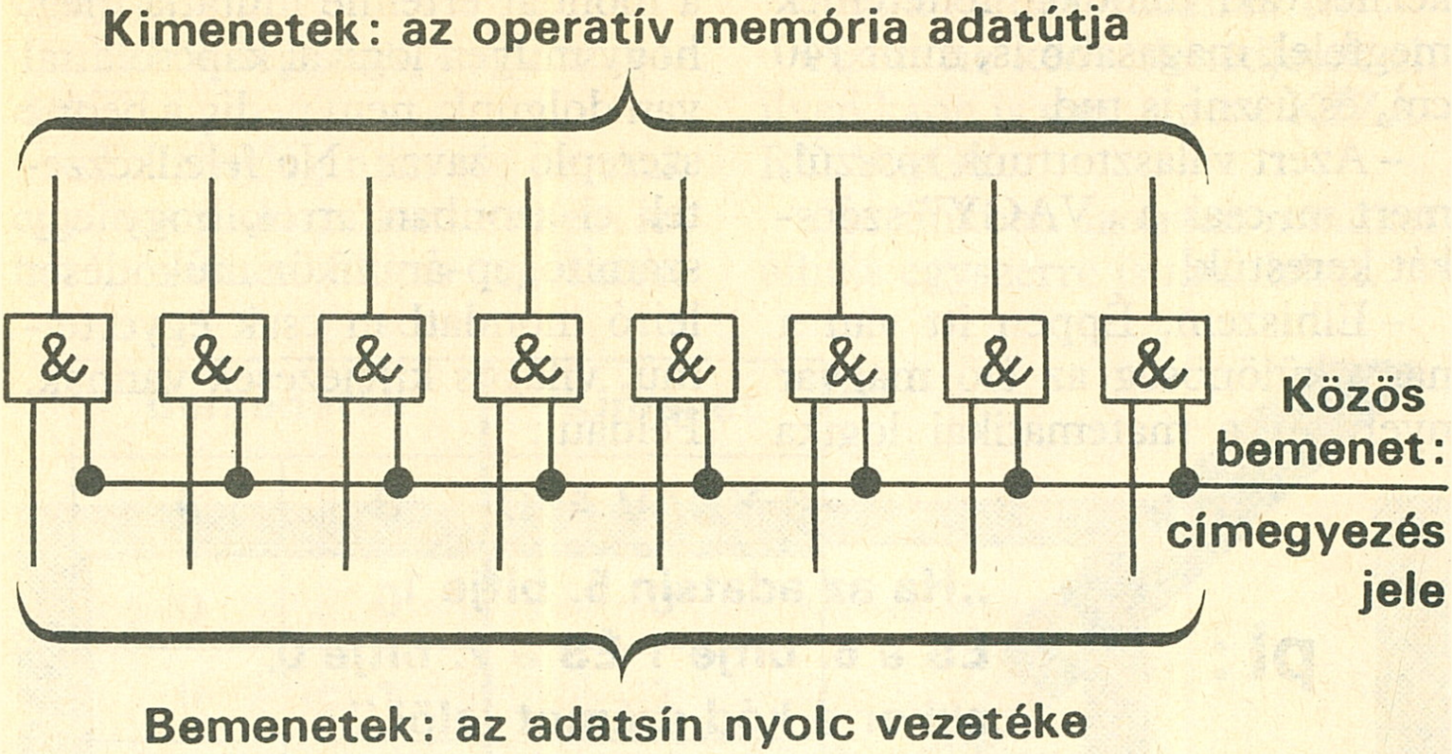
– Éppen ezt akartam elmondani: ilyen egyszerű kapukból lehet összeállítani a tárolóelemeket, a regisztereket, az operatív memória fiókjait szintén, és egészen bonyolult *logikai hálózatokat* is meg lehet építeni.

– Csak ebből a kettőből?

– Csak ÉS meg VAGY kapuból még nem, de ha hozzávesszünk még egy harmadik fajtát, a NEM kaput, akkor már igen.

– Ilyen hálózat van a sorompó helyett is?

– Igen. Elképzelhető egy ilyen is:



Gyerekek! Aki kíváncsi a sorompó működésére, a *Megoldásokban* megtalálja, a 121. oldalon.

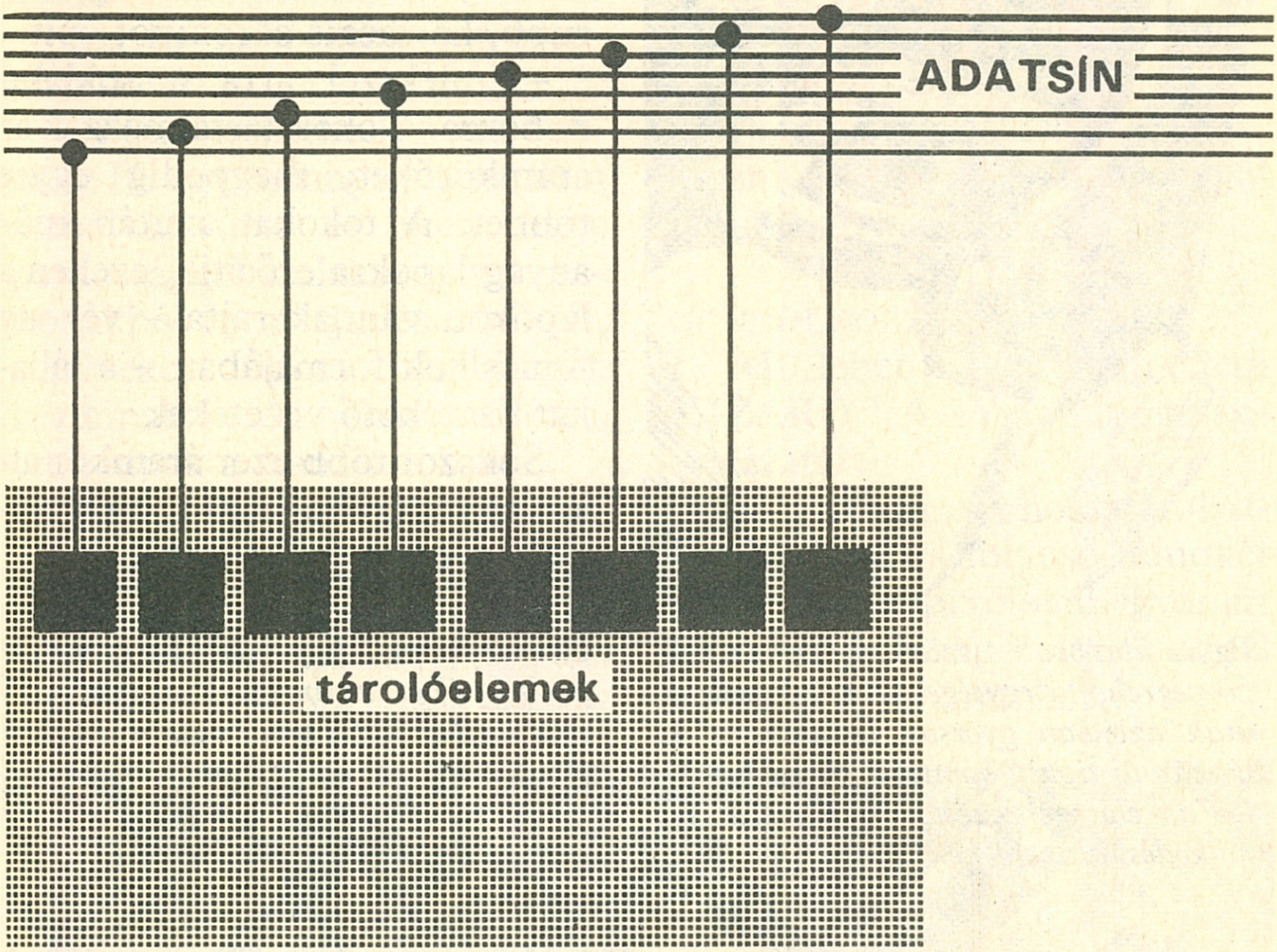
Mi van a fiókban?

– Most figyelj rám, Kinga! A *tárolóelem* egy bit tárolására szolgáló áramkör, és éppen úgy logikai kapukból lehet összeépíteni, mint a számítógép többi áramkörét.

Van ám azonban egy lényeges tulajdonsága: a tárolt bit értékét – legyen az 1 vagy 0, akkor is megőrzi, ha a bemenetéről már rég eltűnt a bemenőjel.

Ilyen tárolóelemekbe futnak be az adatutak minden egyes egységben az adatsínről.

Az egy teljes byte tárolására szolgáló elemsort *regiszter*nek hívják.



– Ilyen regiszterekből áll egy-egy fiók is a memóriában?

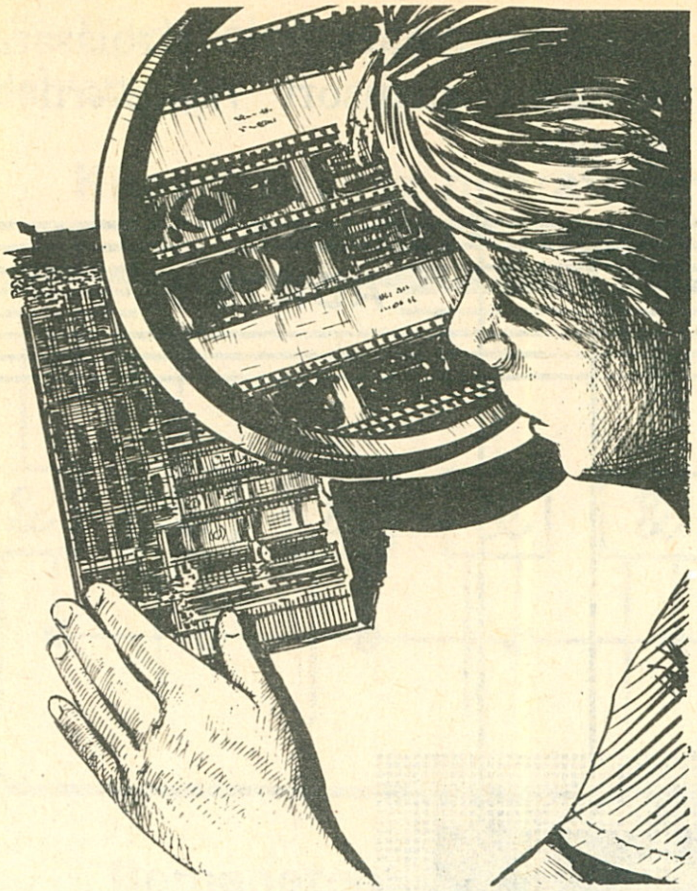
– Nem pontosan ilyenekből, de hasonlókból. Az operatív memória egy-egy rekesze – vagyis fiókja – mindig annyi bit tárolására alkalmas, ahány bites az adatsín. Esetünkben az adatsín 1 byte-os, tehát 1 byte-os méretű egy rekesz is. Ezt a méretet szokták *szóhosszúság*nak is nevezni.

– Sok fiók, akarom mondani, rekesz van a memóriában? – kezdett érdeklődni Balázs.

– A rekeszek számának egysége a *kbyte;* vagy röviden *k*.

1 kbyte = 1024 byte

A kicsi gépek operatív memóriája 16, 32, 48k szokott lenni, ilyenekkel találkozhattok az iskolátokban is, de vannak 1024 kbyte-os gépek is.



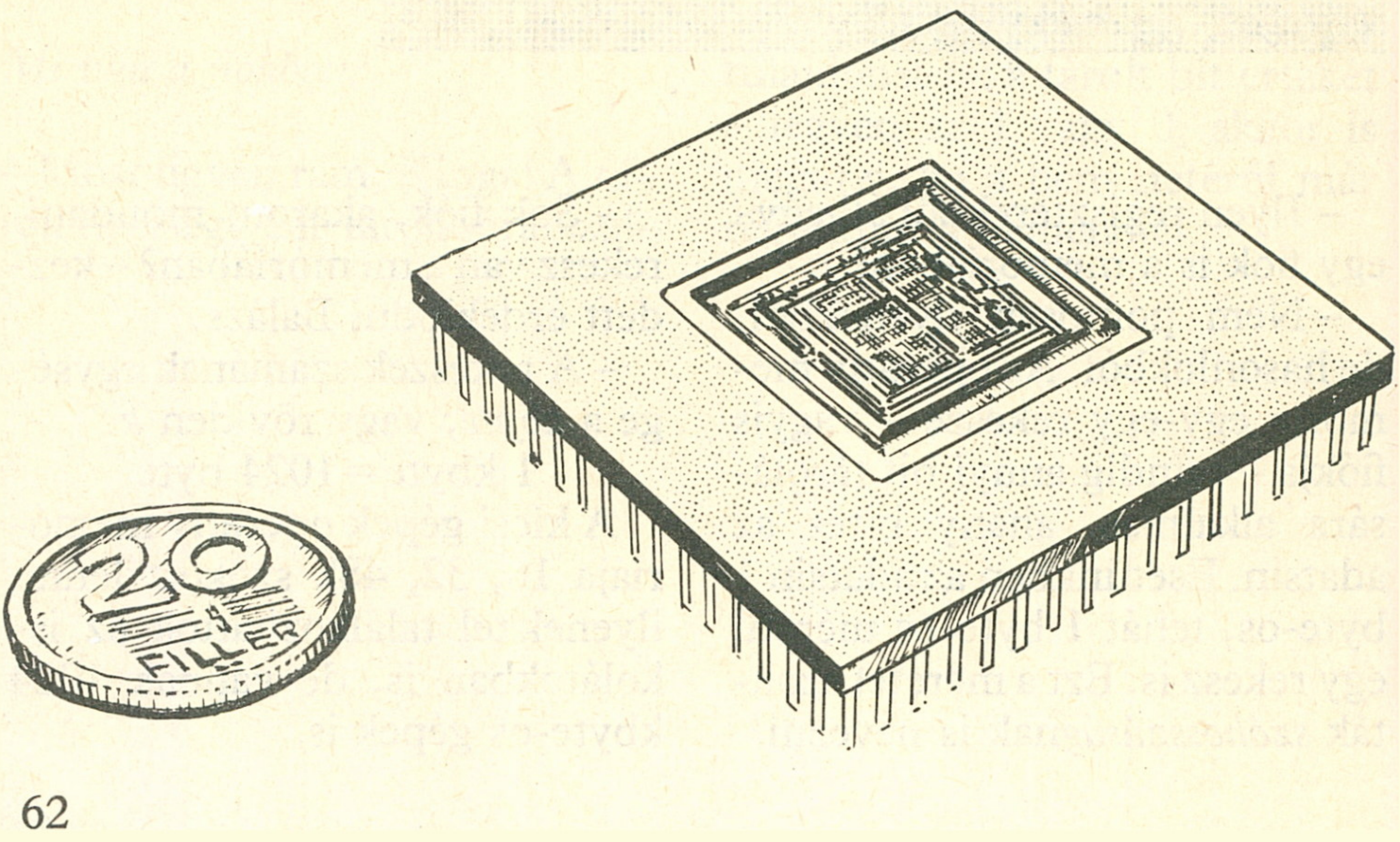
– És milyen a valóságban egy olyan hálózat, mint például az adatsorompó? – ravaszkodott Balázs, mert döntött: egyszerűbb, ha mégis sorompót épít.

– Emlékszel arra a soklábú dobozra? Tokok tartalmazzák az áramköröket, mégpedig egyre többet. A tokokat azután műanyag lapokra erősítik, ezeken a lapokon vannak rajta – vékony fémcsíkok formájában – a lábakat összekötő vezetékek.

Sokszor több ezer áramkör alkot egyetlen hálózatot, mely végül egyetlen tokba kerül.

Egyre apróbbak a számítógépek szervező-végrehajtó egységei. Az információutak azonban gyorsan telítődnek. Az igazán új típusú komputerek valószínűleg az élő szervezettől lesik majd el a működési alapelveket

Szupermorzsa. Egy húszfilléres mellett is eltörpül ez az integrált áramkör. 20 ezer logikai kapu van benne – százszor több, mint az eddig használt hasonló mikroáramkörökben, mindössze 1,2 négyzetcentiméternyi felületen



Az ilyen tokot *IC*-nek, integrált áramkörnek hívják, de kis mérete miatt *chip*nek, azaz morzsának is.

– És ha mikroszkópban nézném meg?

– Akkor láthatnád, hogy bármely IC egy műanyag lapocskán szabályszerűen és igen nagy pontossággal kialakított kis csatornákból és mélyedésekből áll, amelyeket finom fémrétegek borítanak be. Ahol köröket látni, ott vannak a tranzisztorok. Egy ilyen morzsából csak a lábak, a kivezetések állnak ki, nem is lehet szétszedni őket.

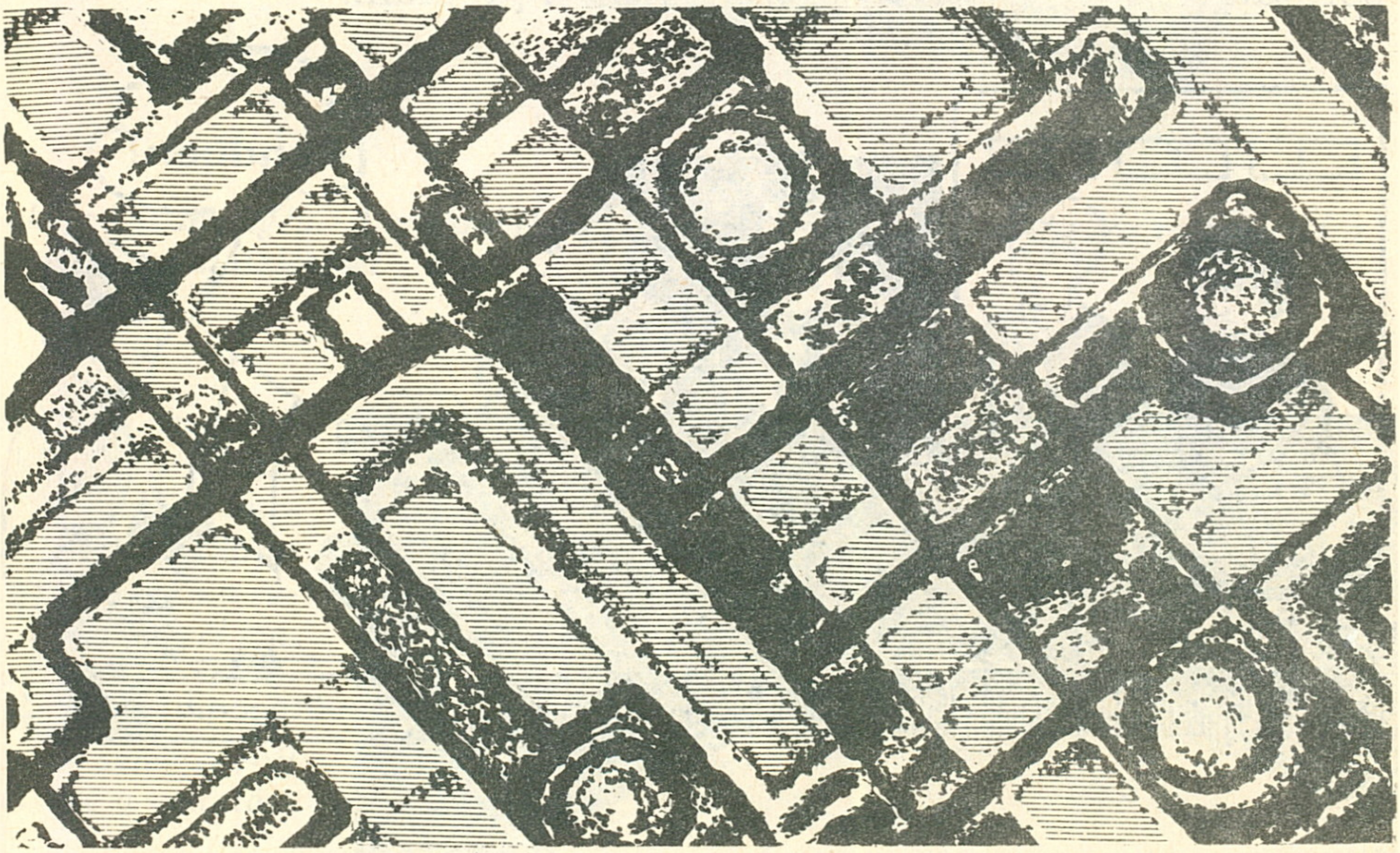
– Kár. Akkor én most hogy építsek?

– Sehogy, Balázs. Nemsokára meglátod, hogyan lehet programozni, és azzal is nagyon jól fogsz szórakozni.

GONDOLKOZZATOK!

1. Mi lehet egy kétbemenetű KIZÁRÓ VAGY kapu igazságtáblázata?
2. Házunkban egy hosszú folyosó világítását a folyosó mindkét végén lévő kapcsolóval egyaránt lehet be- és kikapcsolni. Tudnál tervezni ilyen áramkört?

Akusztikai mikroszkóppal készült felvétel. Az integrált áramkör élesen kirajzolódó felületi képe és finoman árnyalt belső szerkezete egymásra vetülve látható



Mi is a program?

– Tudjátok-e, mi is az a *program*?

– Hát persze! Az őrsben is szoktunk programot csinálni – kezdte Kinga. – Megbeszéljük, hogy ki, hová akar menni szombaton délelőtt, azután eldöntjük, hogy mit is fogunk csinálni. Ha erről lesz szó, én már tudom!

– Nem, nem, egészen másról. A program szó több jelentésű. Az egyik jelentését már ismered, a másik jelentése pedig utasítássorozat.

– Ez valami új dolog?

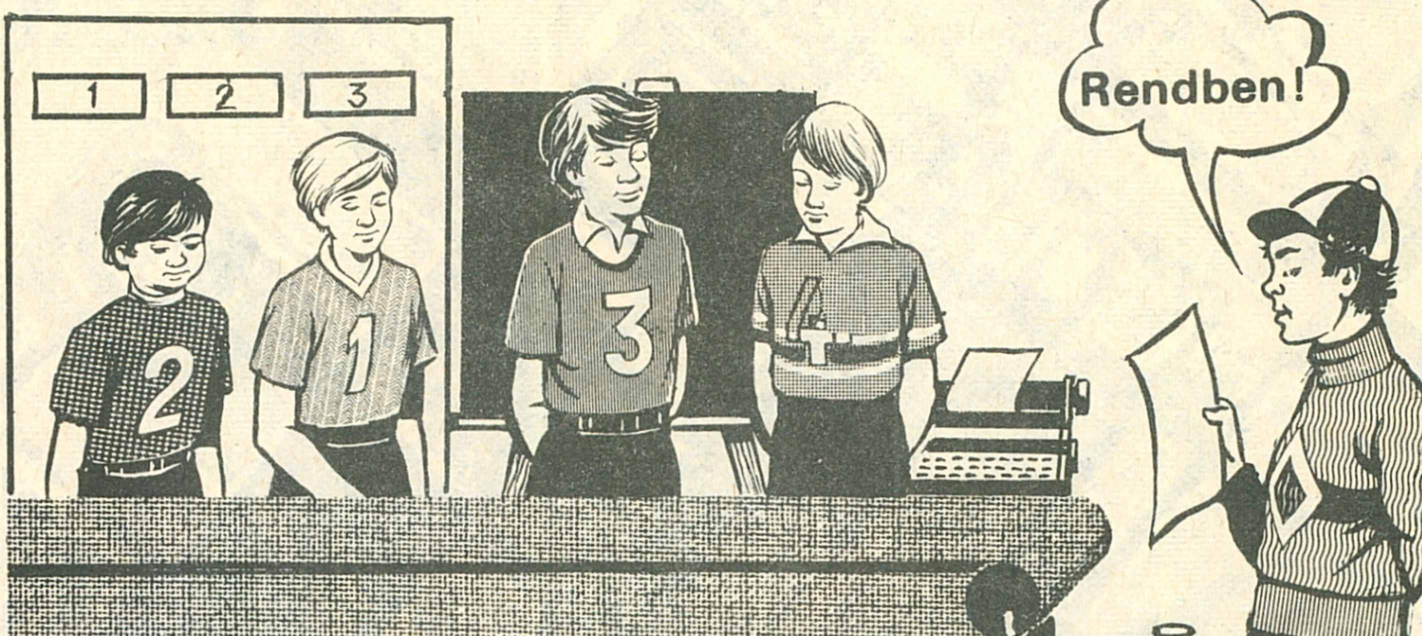
– Nem, már ismeritek, csak nem így hívtuk eddig. Emlékezzetek, amikor a 8-ból kivontuk a 2-t, így kezdődött a munka:

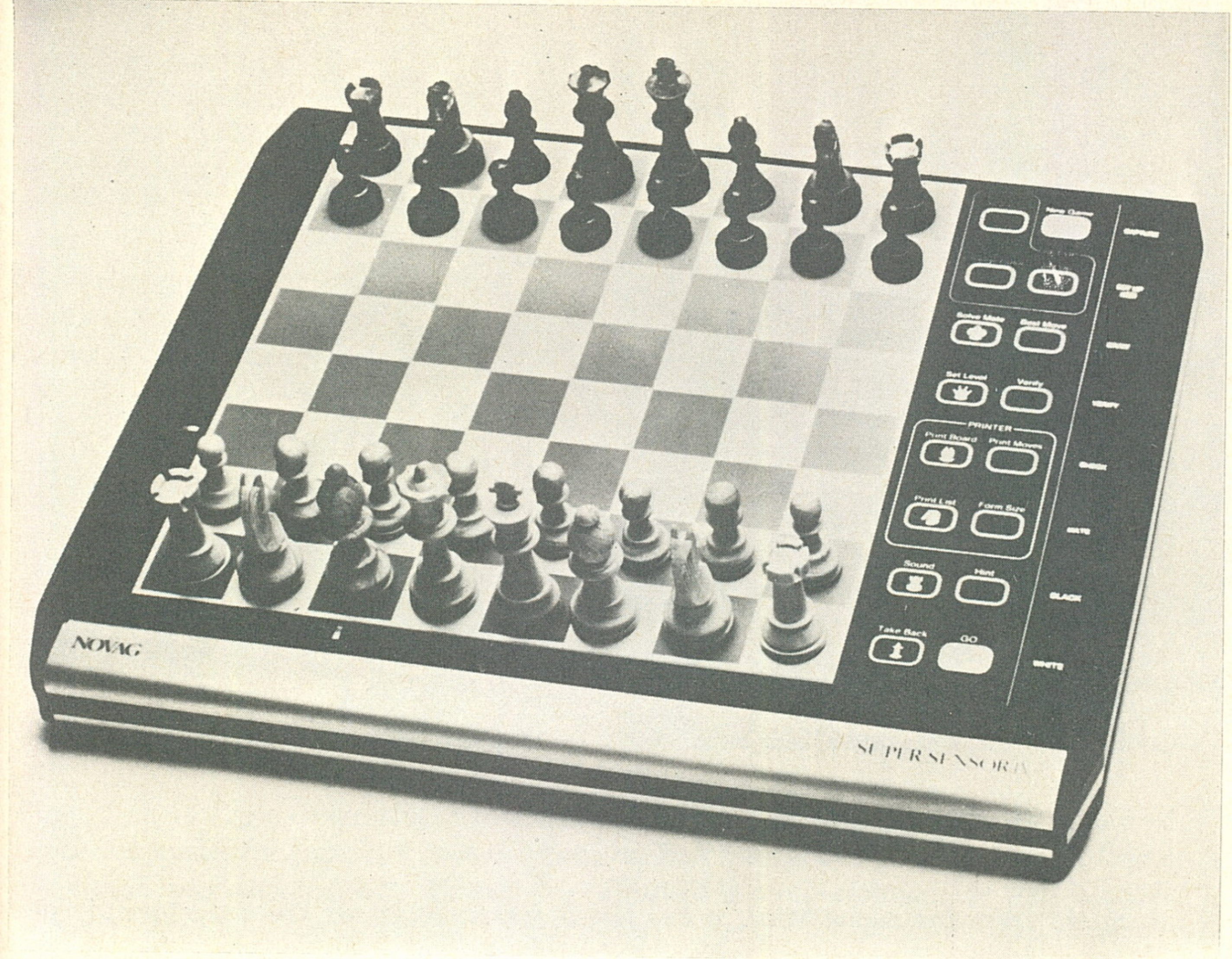
„Pista I, 1-es ki!" – vezényel Főpista.

Az 1-es fiókban lévő csomag a szállítószalagon Főpista elé kerül, aki kibontja, és elolvassa a tartalmát:

„A 2-es fiók tartalmából ki kell vonni a 3-as fiók tartalmát, s az eredményt leírni."

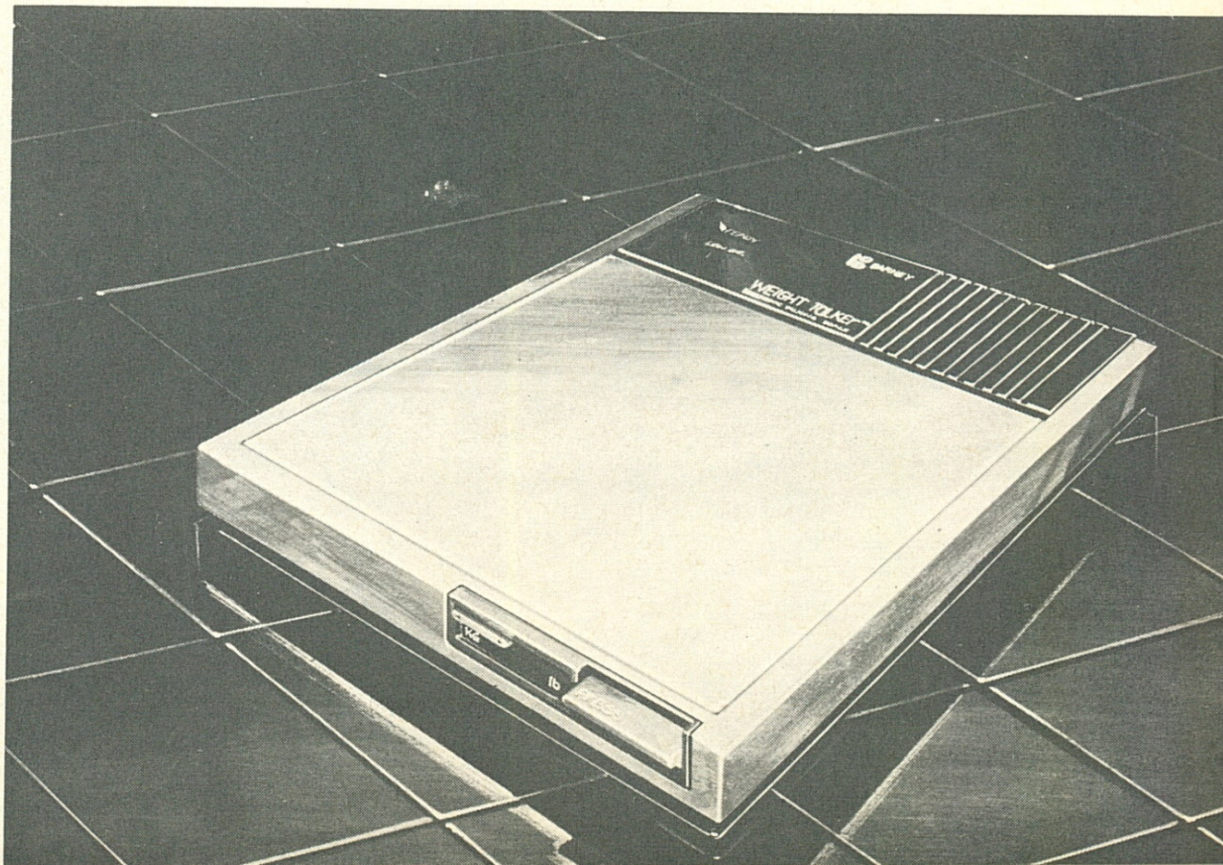
– Amit Főpista olvasott, az volt a program?





Sakkozó számítógép. A sakktábla megfelelő mezőjén megjelenő figura rajza jelzi a számítógép válaszlépését

Beszélő személymérleg. Ha valaki rááll, beszédszintetizátorral előállított férfihang közli a súlyát. Bizony jól jön, ha az illető a pocakjától nem látja, melyik számjegynél állt meg a mutató

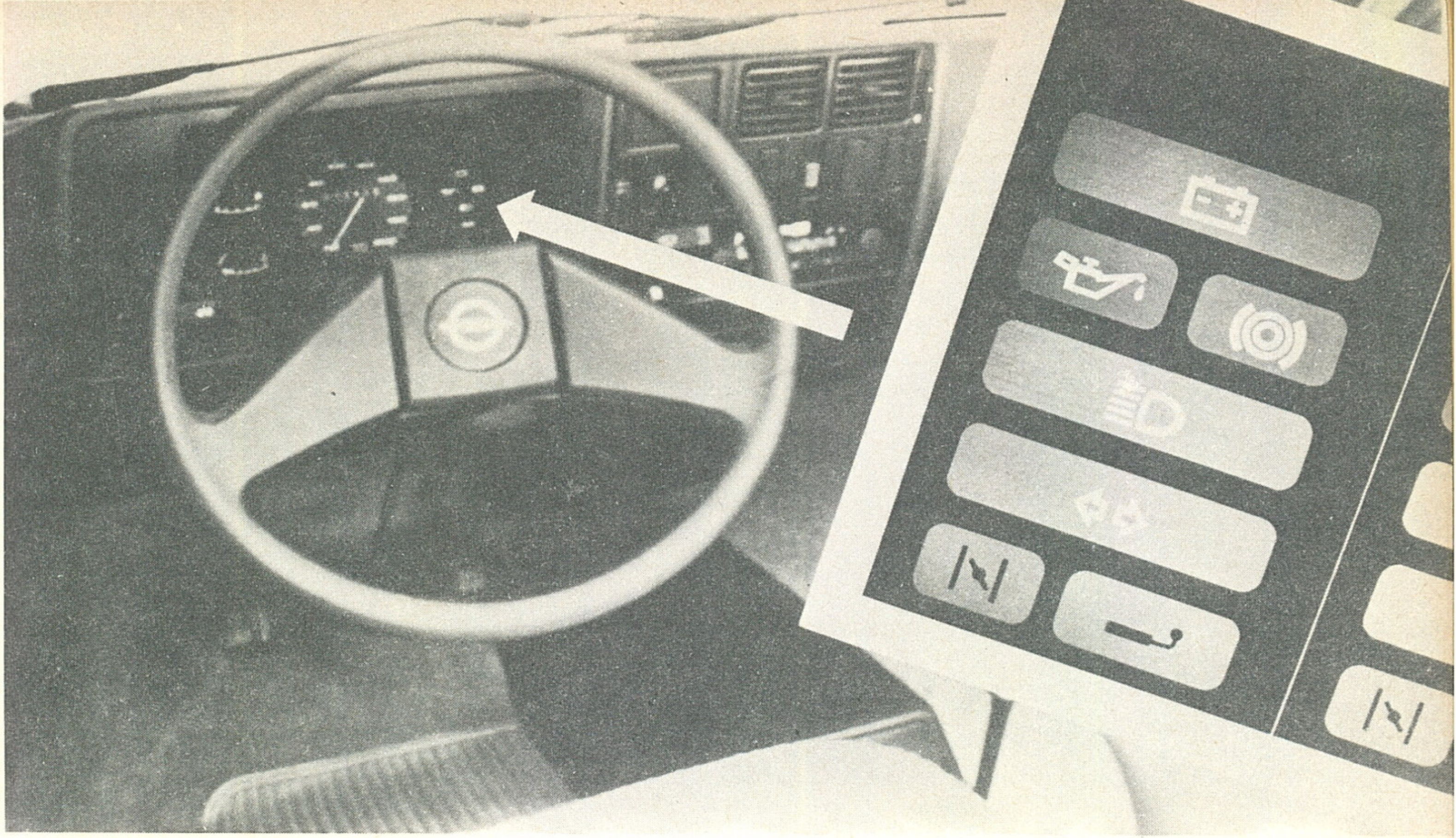




Kis számítógép gondoskodik arról, hogy az egyes üvegházakban termesztett növények megkapják mindazokat a feltételeket, amelyekre legkedvezőbb fejlődésükhöz szükségük van: nemcsak a megfelelő hőmérsékletet és szellőzést, hanem például az igényeik szerinti légnedvességet, talajhőmérsékletet, fénysugárzást, szén-dioxidot stb.

Az üvegház légnedvességi adatait a bal oldali mérőberendezés küldi a számítógépbe





Könnyebben leolvasható visszajelzők. Helyüket a nyíl mutatja a műszerfalon

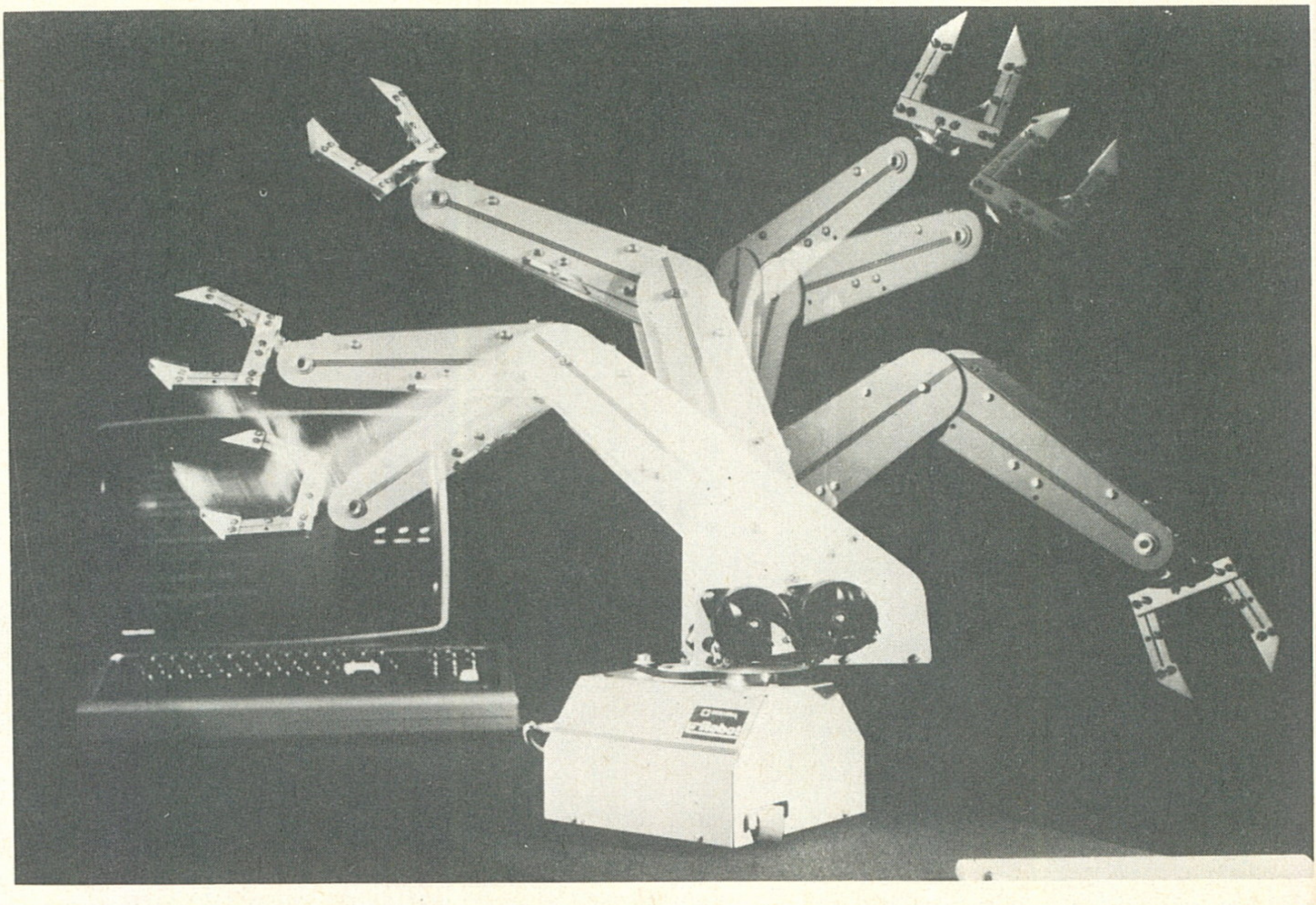
Bizony jól jön egy ismeretlen városban, ha a képen látható kinagyított „szélrózsa" felvillanó nyilai fokozatosan mutatják, merre kell haladnunk, hogy a legrövidebb úton közelítsük meg úticélunkat. Ennek koordinátáit persze előzőleg be kell táplálni a gépkocsi „pilóta"-számítógépébe. A többi már – persze a haladási irány és a megtett út folyamatos észleléséből nyert adatok alapján – az ő dolga

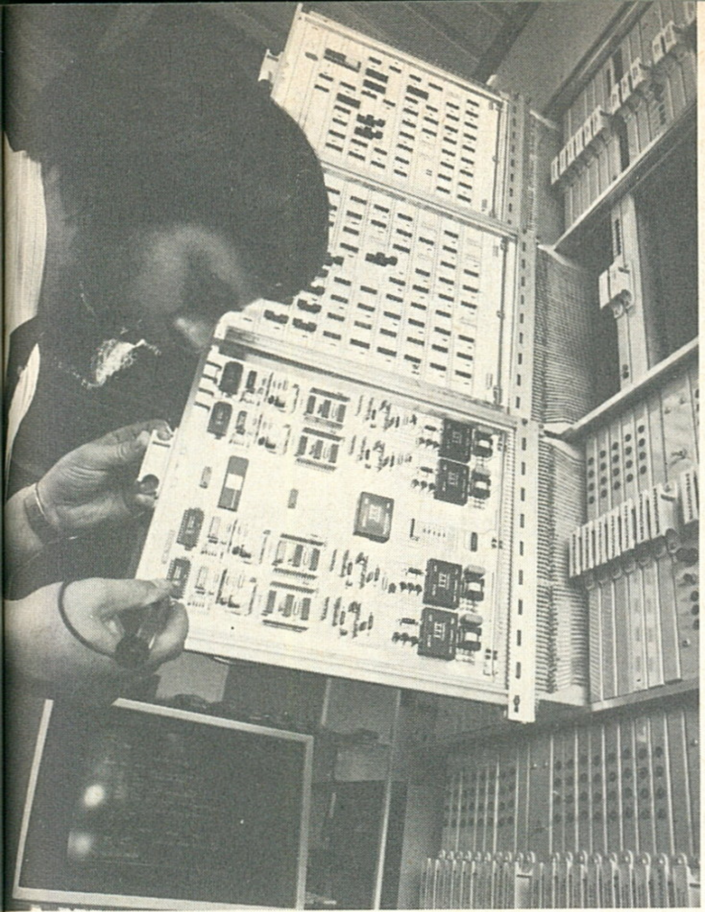


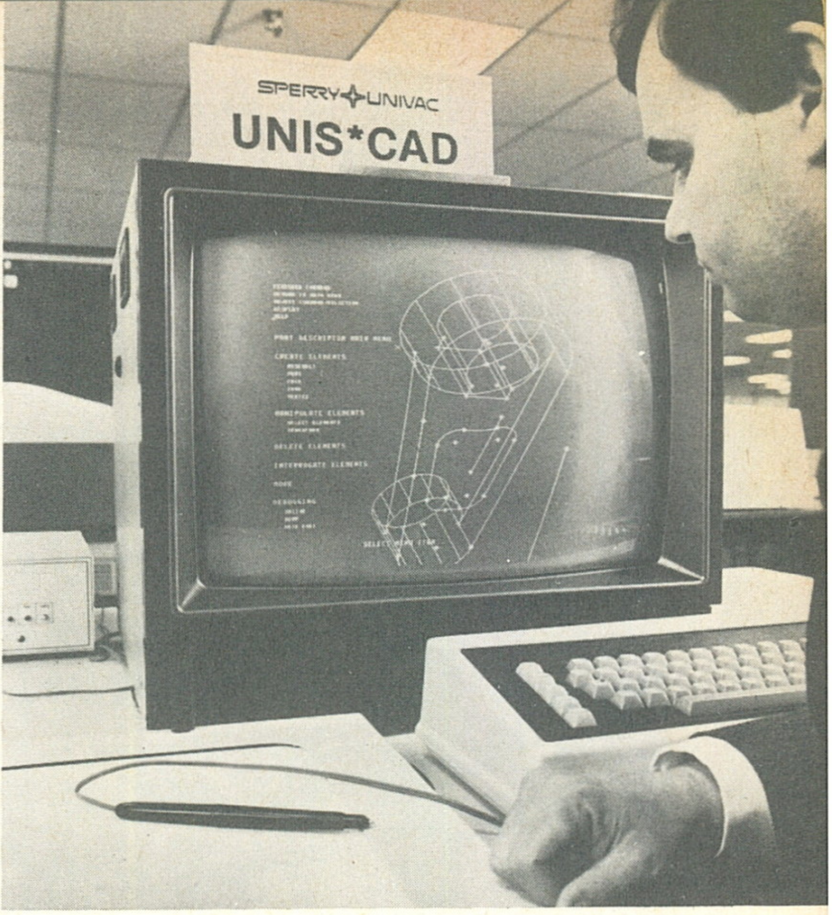


Az autópálya forgalmát is számítógép ellenőrzi és irányítja. Számba veszi az egyes sávokban haladó autók számát és sebességét, és ennek alapján az út felett elhelyezett kijelzőkön közli a mögöttük elhelyezkedő útszakaszon haladókkal az optimális sebességet

Ez a különféle tárgyak, alkatrészek mozgatására szolgáló robot a kép bal sarkában látható számítógép segítségével sokféle munkára programozható

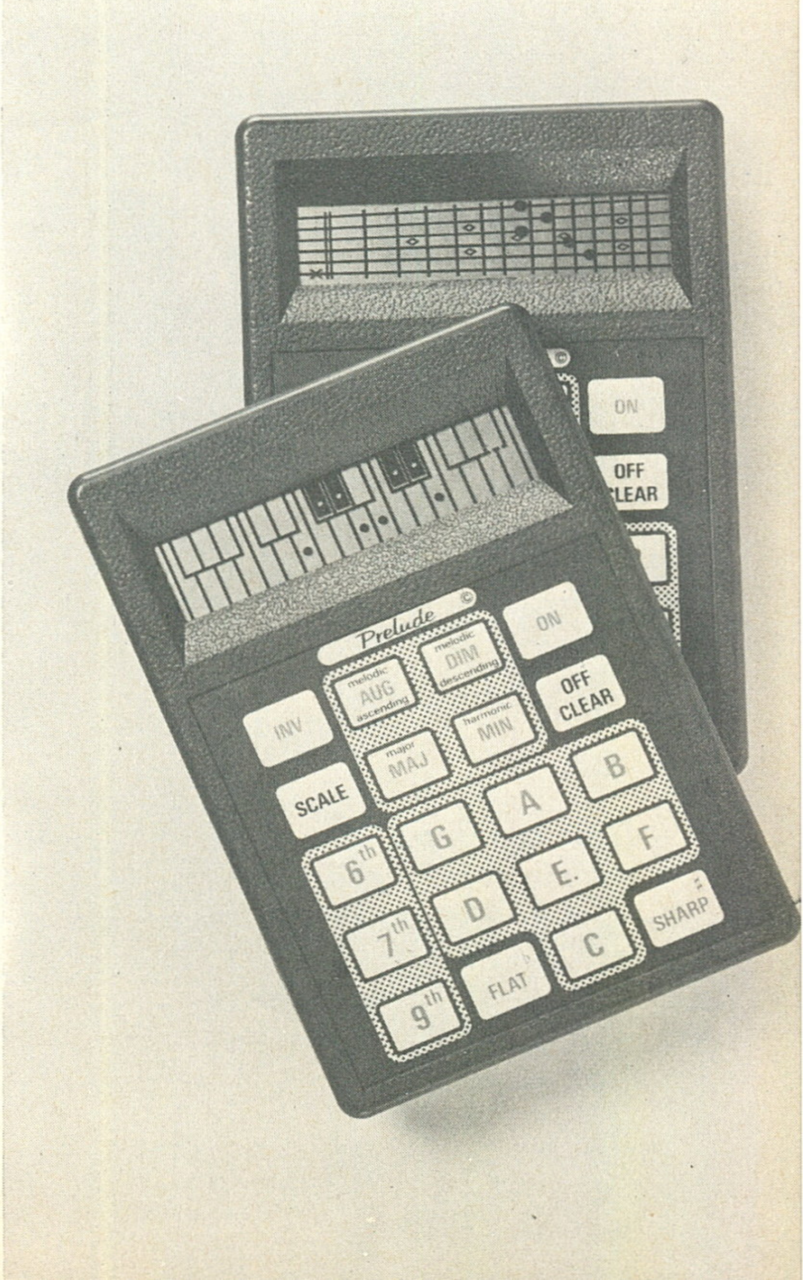






Mikroelektronikai elemekből felépített, digitális technikával működő telefonközpont

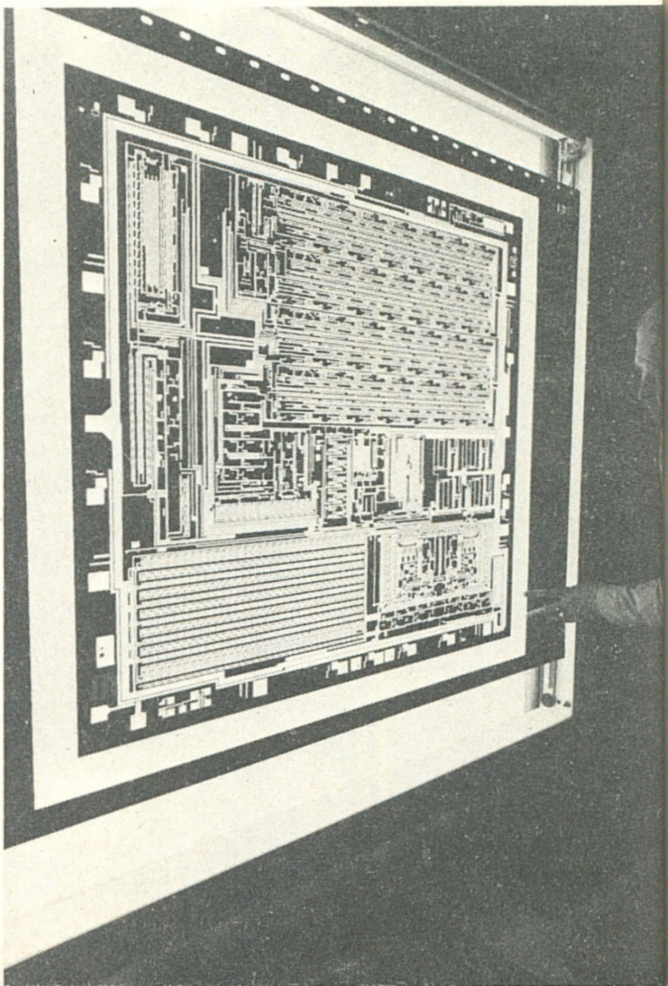
Az UNIS CAD számítógép a betáplált adatokból elkészíti és a képernyőn megjeleníti a grafikus ábrát a mérnök számára. Ezen azután tetszése szerint módosíthat, és nyomban szemmel is tanulmányozhatja a változtatás eredményét (jobbra fent)



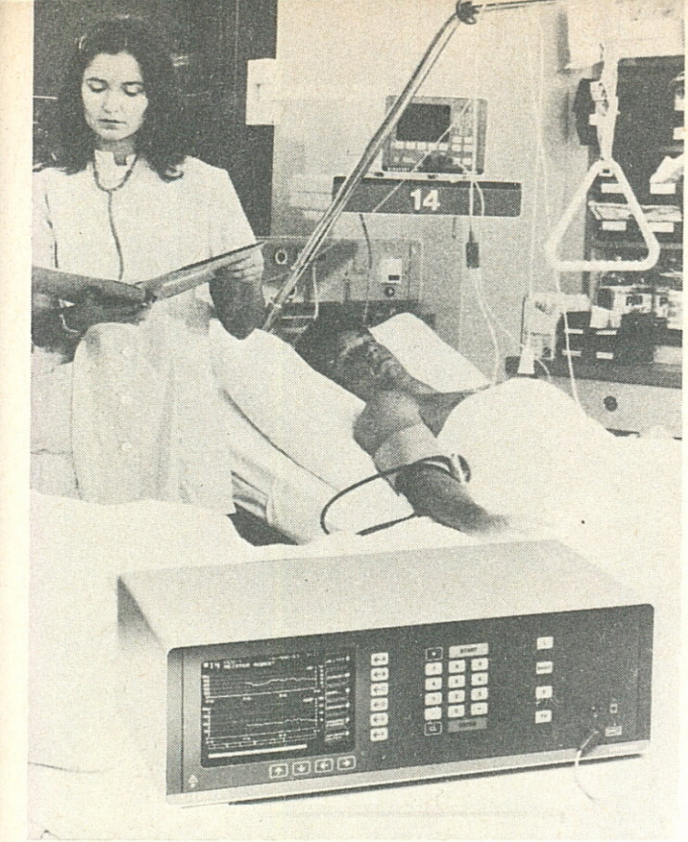
Ez a kis zsebszámítógép egy adott dallamhoz megfelelő kíséretet tud komponálni. Miután a zongorabillentyűkön betáplálták a dallamot, a kottavonalakon felvillanó fénypontokkal jelzi, mely hangokat kell megszólaltatni a hangszeren



A chipek különféle parányi elemeit csak ilyen vetítőernyős nagyítókészülék segítségével vizsgálhatják

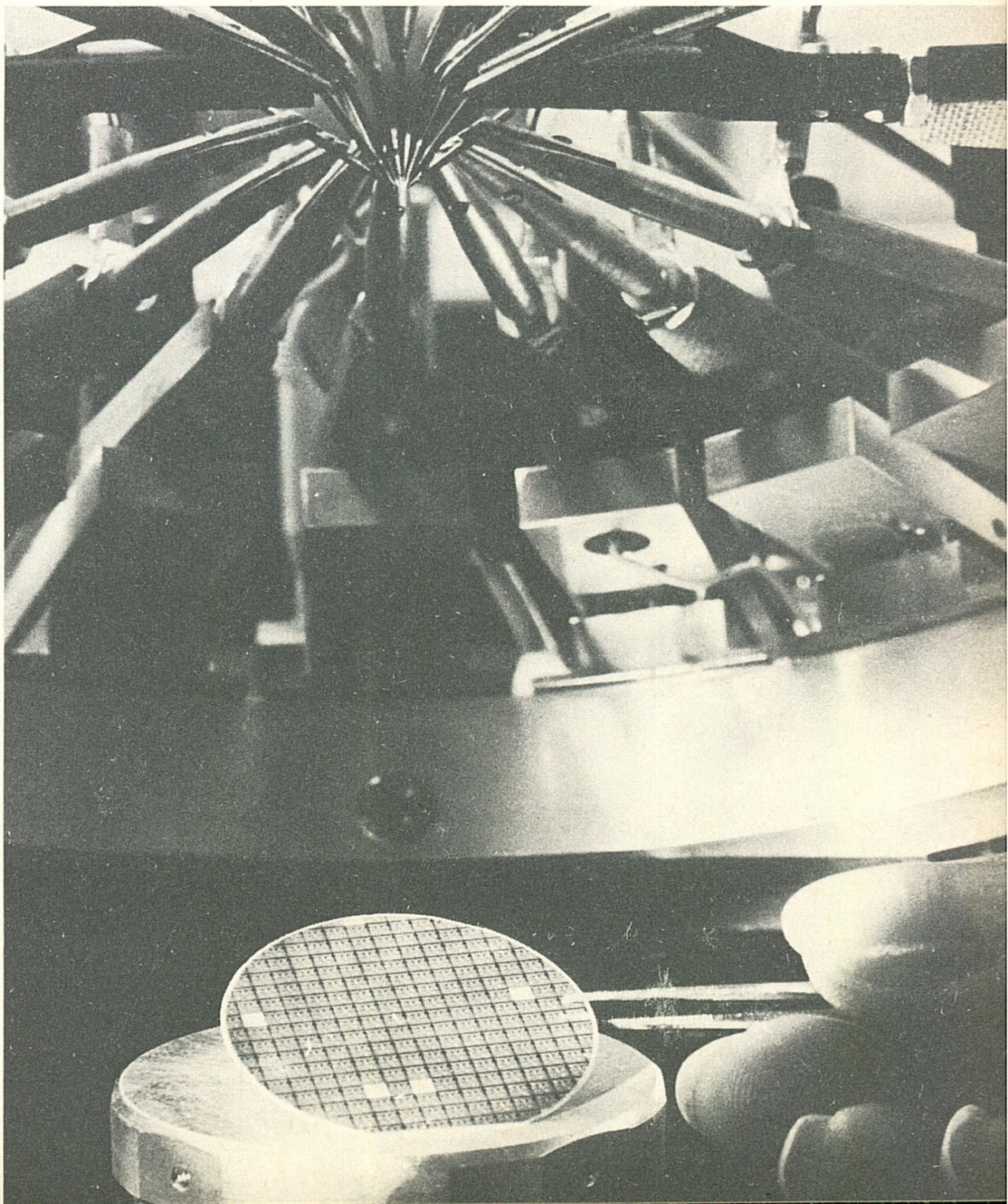


Ilyen egy chip óriásira nagyított tervrajza. Ha majd legyártják, egyetlen négyzetmilliméterén 250 000 kapcsolási elemet tartalmaz

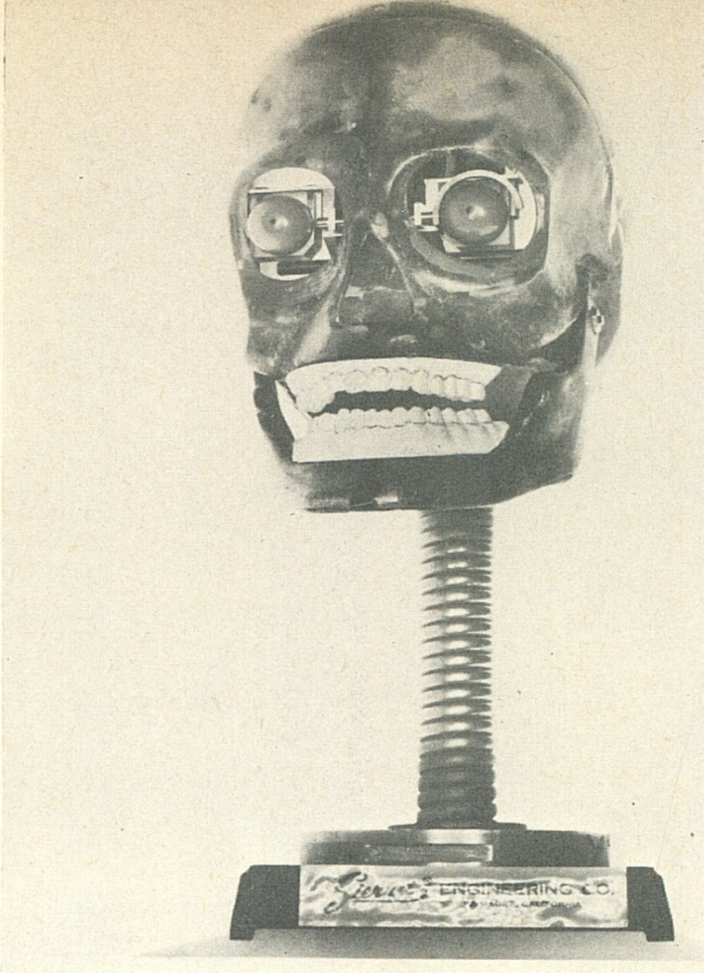


Számítógéphez kapcsolt ellenőrző műszerek figyelik a beteg állapotát. Ha például hőmérséklete, vérnyomása, szívműködése eltér a számítógépbe táplált megengedett értéktől, a készülék rögtön riasztja az orvosokat

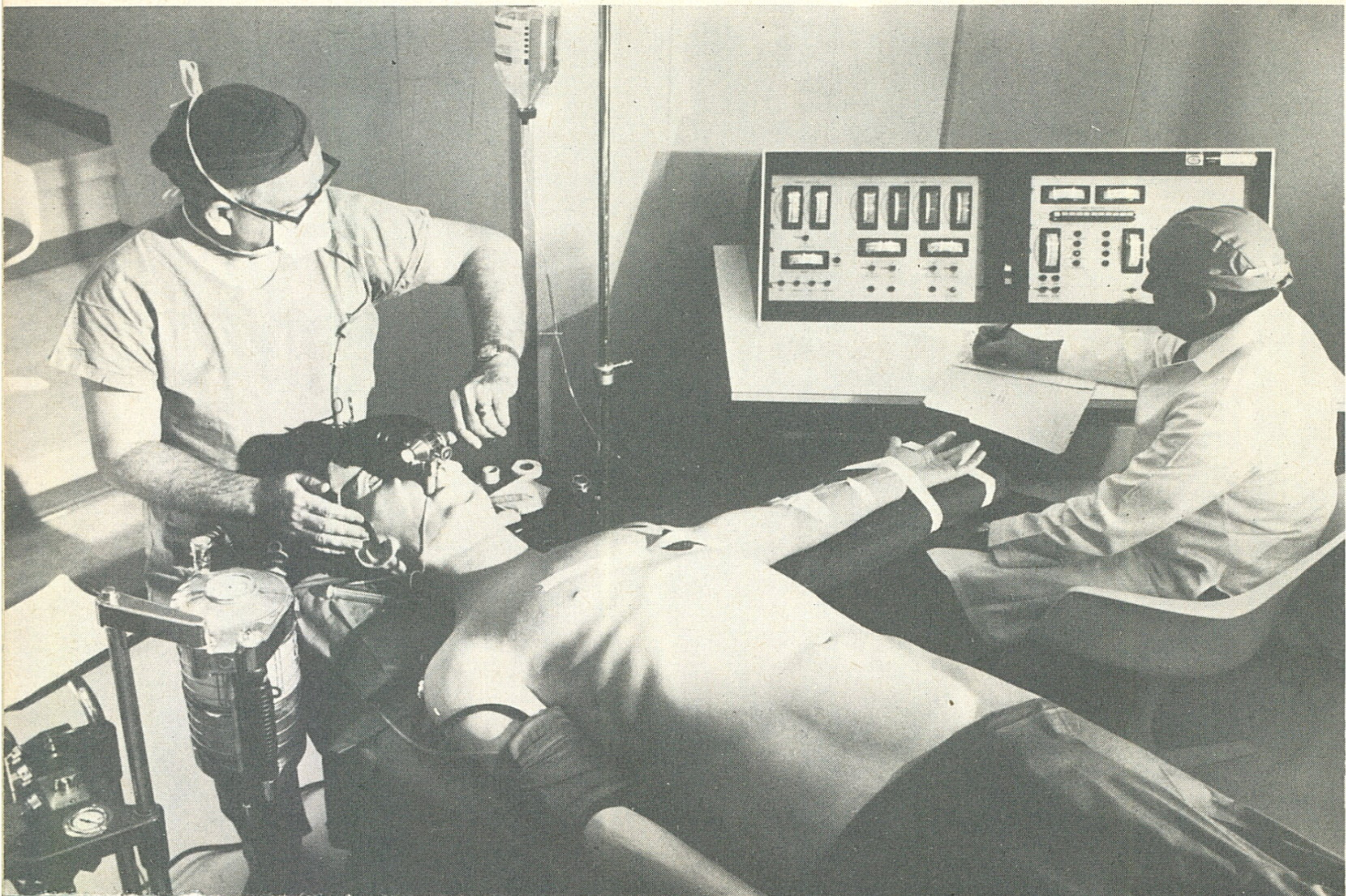
Egy-egy két milliméter élhosszú szilícium lapocskán sok száz tranzisztor, dióda, ellenállás helyezkedik el. Egyszerre sok-sok ilyen lapocskát helyeznek el az ellenőrző készülékbe, hiszen valamennyi lapocska valamennyi elemének megbízhatóan kell működnie



Ez a számítógéphez kapcsolt robot nemcsak pislogni, lélegzeni, köhögni vagy hányni tud, hanem akár meg is halhat, illetve mindezt utánozhatja a számítógép parancsára, ha az őt kezelő és a műtéti altatást rajta gyakorló orvostanhallgató hibásan végzi a dolgát



Ez is robot, pedig látszólag olyan, mintha élő ember lenne. Szintén az altatás begyakorlására szolgál. Itt azonban a „beteg" pulzusának, vérnyomásának és más „élettani" funkcióinak adatai a hozzákapcsolt számítógép kijelzőin jelennek meg



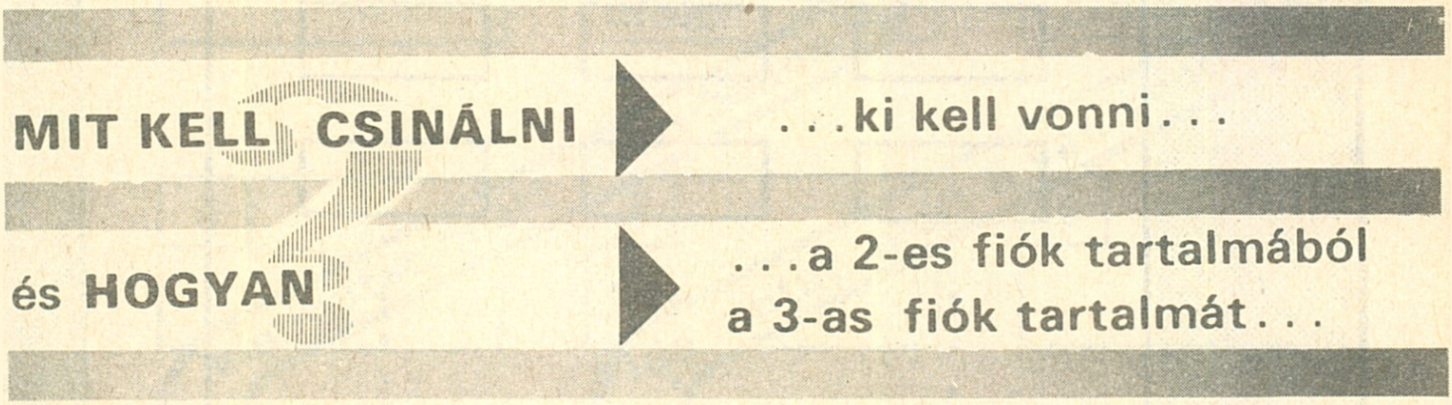
– Igen. Most, hogy már egy kicsit jobban ismeritek az adatábrázolást meg az adatáramlást, megmutatom, milyen is pontosabban ez az utasítássorozat.

A memória rekeszeinek hossza 1 byte, tehát egy nyolcbites kód fér bele.

Mit is olvasott Főpista:

„A 2-es fiók tartalmából ki kell vonni a 3-as fiók tartalmát, s az eredményt ki kell írni!”

Ebben benne van, hogy

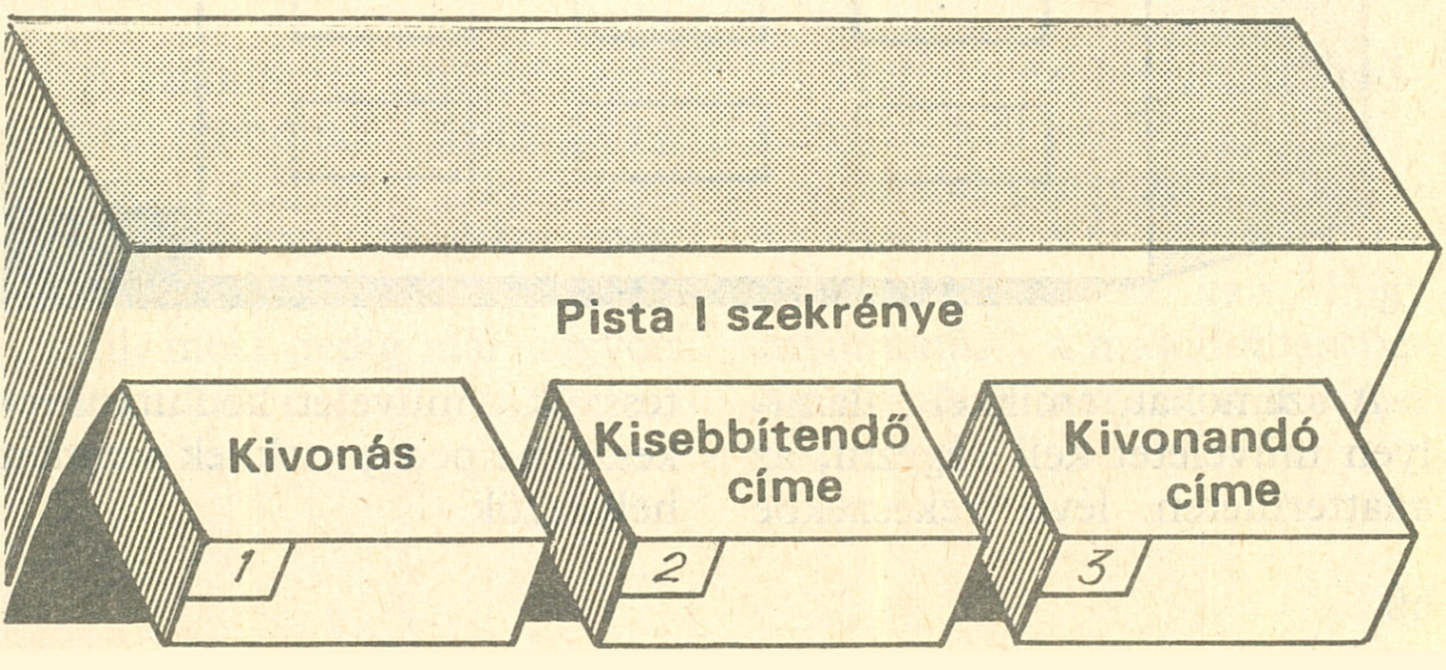


– Az is benne van, hogy az eredményt ki kell írni! – nyújtózkodott Balázs.

– Igaz. Benne van azonban az is, hogy a különbséget hová kell majd elhelyezni. Mi maradjunk abban, hogy a kivonás eredménye a kisebbítendő helyére kerül majd vissza.

– A tennivaló és a módja mind egyetlen 8 bites kódba van beleírva?

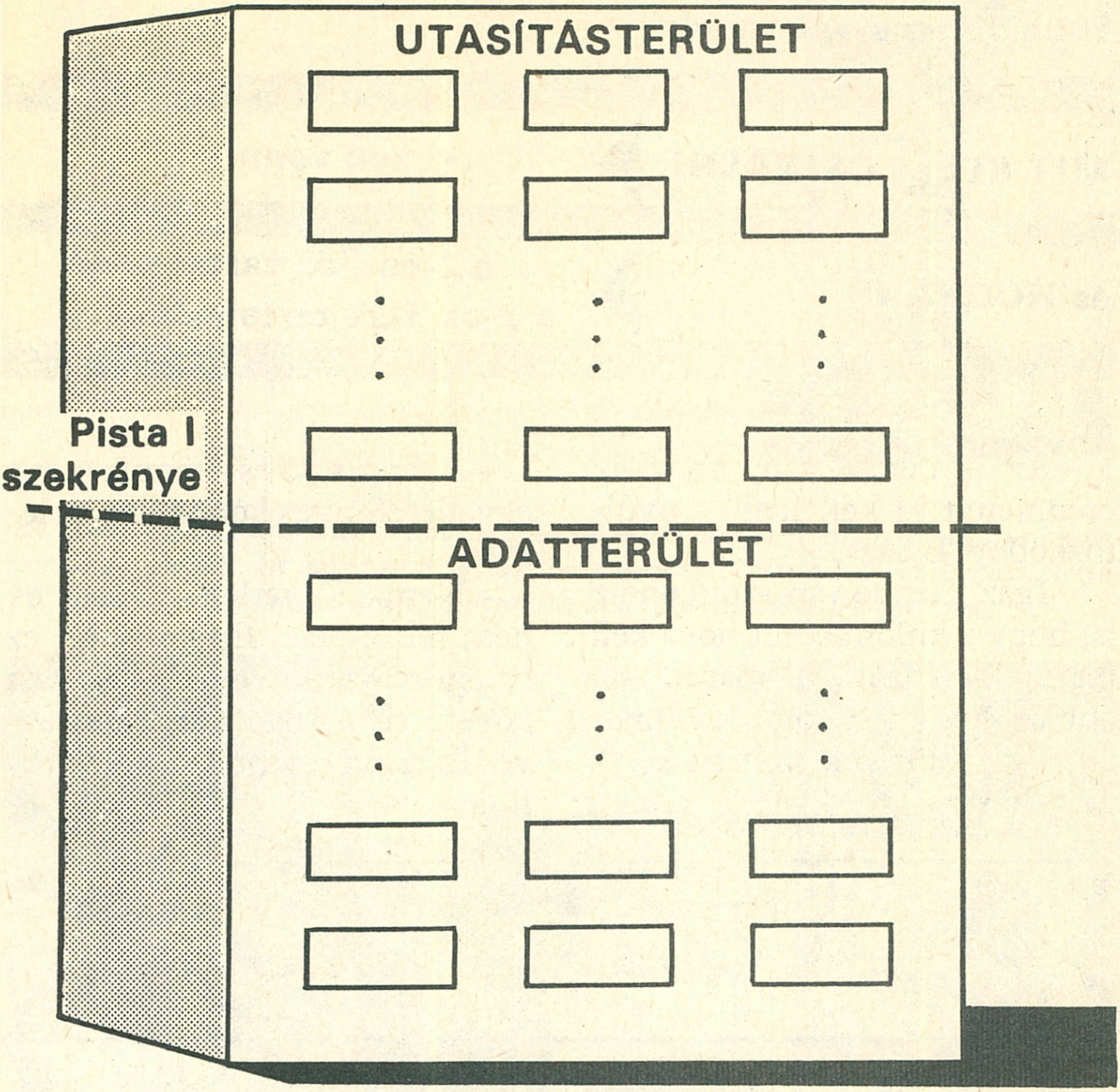
– Nem. Egyetlen kódba ez nem férne bele. Éppen ezért ez az egész üzenet három egymást követő rekeszben van elhelyezve. Ez a programunk első *utasítása*.



Az első rekeszben van a tennivaló, azaz a művelet, amit el kell végezni. Ezt *műveleti kód*nak hívják.

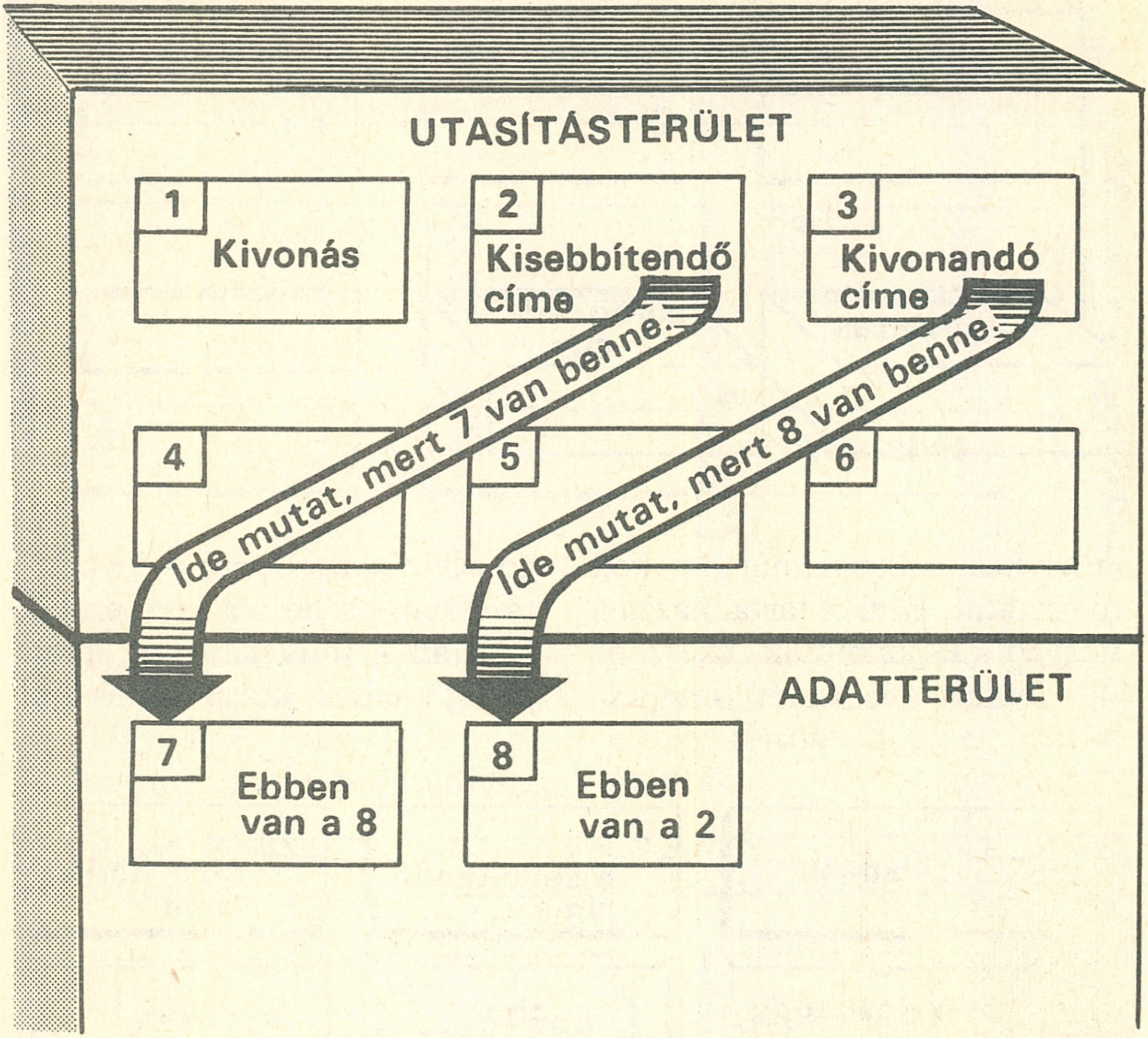
A következő két rekeszben pedig azoknak a rekeszeknek a címei vannak, amelyekben a 8 és a 2 van.

Sokkal egyszerűbb ugyanis a gép működése, ha a memóriát képzeletben két részre osztjuk: *utasításterület*re és *adatterület*re.



A számokat, amikkel valamilyen műveletet kell végezni, az adatterületen lévő rekeszekbe tesszük, a műveleti kód utáni rekeszekbe pedig ezeknek a címeit helyezzük.

A mi esetünkben ez így alakul:

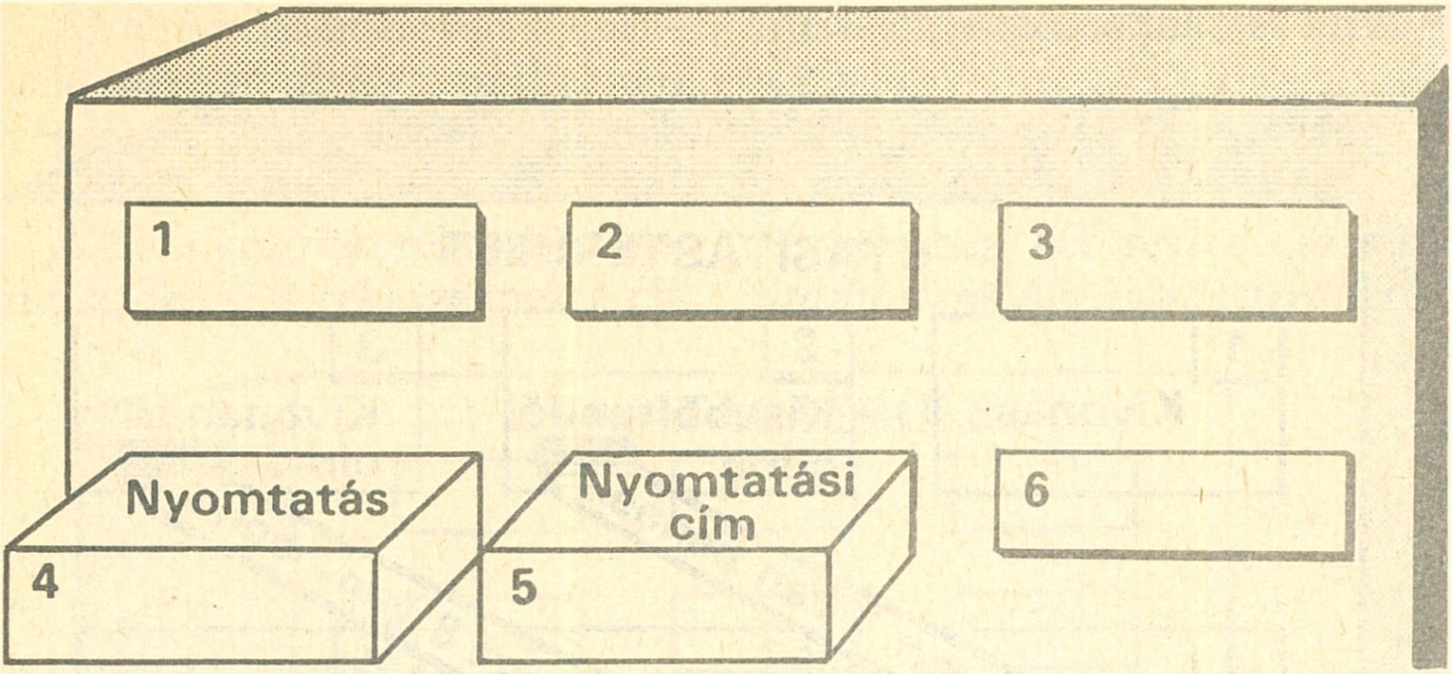


– A számítógép működésénél nem ezt mondtad! – duzzogott a lányom.

– Igazad van, de akkor még semmit sem tudtatok a számítógépről, most pedig már nagyon sokat tudtok.

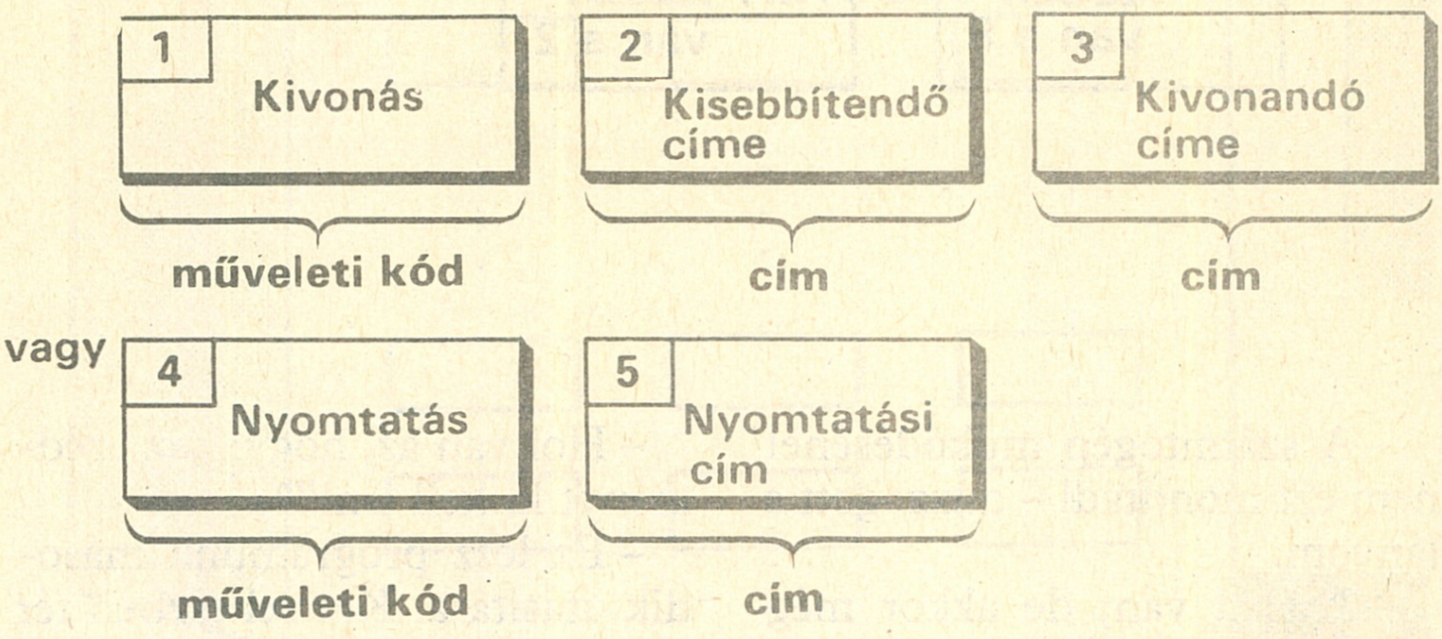
– Hol van az, hogy: „az eredményt ki kell írni”?

– Ez lesz programunk második utasítása. Két rekeszben fér el, az elsőben az van, hogy „nyomtatás”, a másodikban pedig annak a rekesznek a címe,



amelynek a tartalmát ki kell nyomtatni. Ezeket tartalmazza a negyedik és az ötödik rekesz.

– Tudom a lényegét! – ragyogott fel Balázs szeme. – Az utasításoknak hasonló a szerkezetük. Elöl áll a műveleti kód, utána pedig a címek következnek:



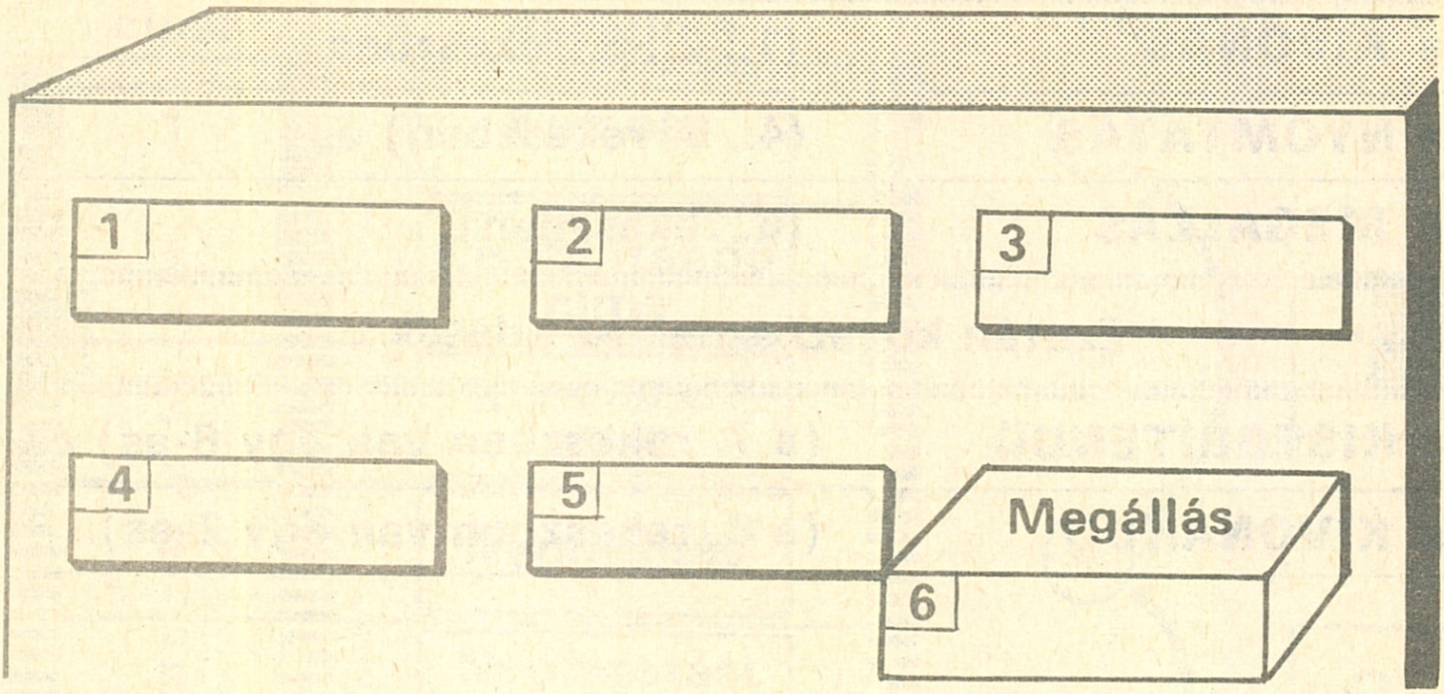
– Ezt jól eltaláltad, örülök.

– Olyan utasítás is van, aminek nincs címe? – kérdezte Csaba.

– De buta vagy! Mire lenne az jó? – így Kinga.

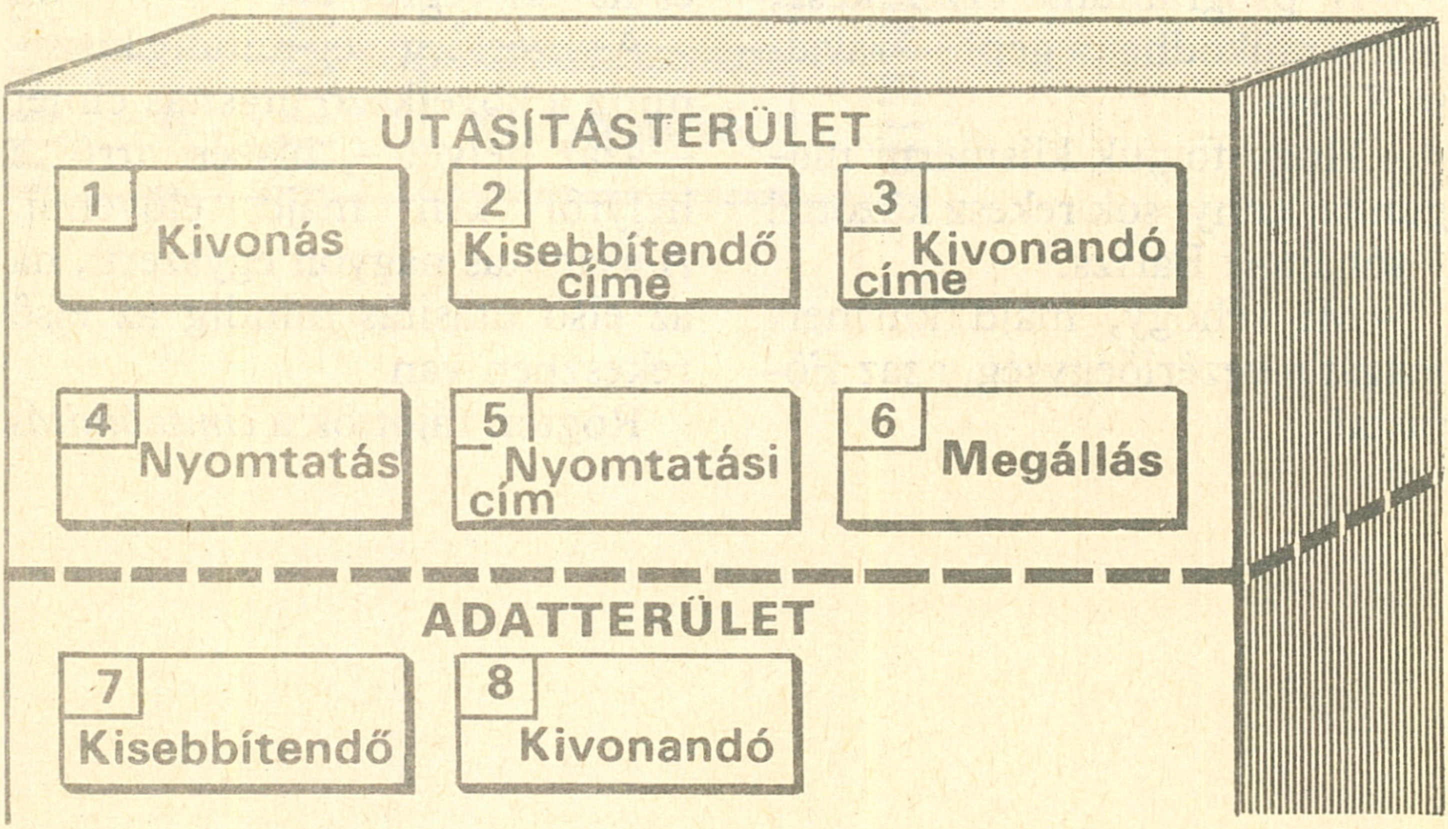
– Ne légy szeles, lányom! Van ilyen utasítás, mindjárt használ-

juk is. Ez a megállító utasítás, ennek elég egyetlen rekesz. Ez kerül a 6-os fiókba.



– A 8-asnak és a 2-esnek is szükséges egy-egy rekesz!

– Úgy van. Összesen nyolc rekeszt tölt meg a programunk, hogy könnyebb legyen áttekinteni, minden egyes fiók elejére ráírom, hogy mit tartalmaznak:



Három **UTASÍTÁS** követi itt egymást:	
a **KIVONÁS**	(1., 2., 3. rekeszben),
a **NYOMTATÁS**	(4., 5. rekeszben) és
a **MEGÁLLÁS**	(6. rekeszben).
Ezután következnek az adatok:	
a **KISEBBÍTENDŐ**	(a 7. rekeszben van egy 8-as) és
a **KIVONANDÓ**	(a 8. rekeszben van egy 2-es).

– És hol az eredmény, a különbség?

– Az a működés után a 7. rekeszben lesz (ahol a kisebbítendő volt).

– A programunk ezzel kész, máris működhet a gép! – szavalta Kinga.

– Hogy fogjuk kiismerni magunkat ennyi sok rekesz között?! – sóhajtott Balázs.

– Mi sehogy, majd kiismeri magát a vezérlőegység, azaz Főpista! *Egy igazi automata!*

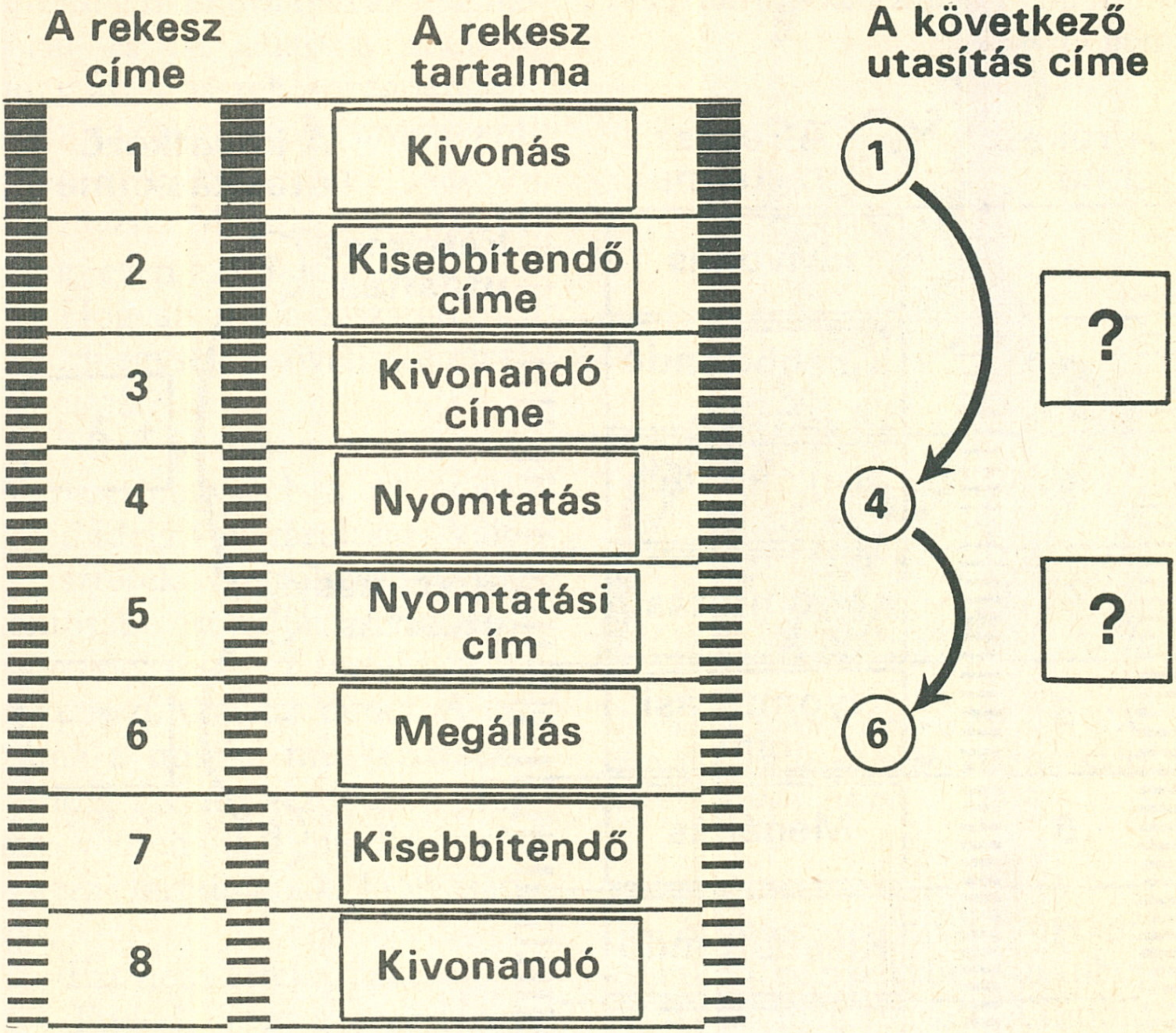
Főpista tudománya ebben az egyetlenegy mondatban áll:

„Vedd a következő utasítást, és hajtsd végre!”

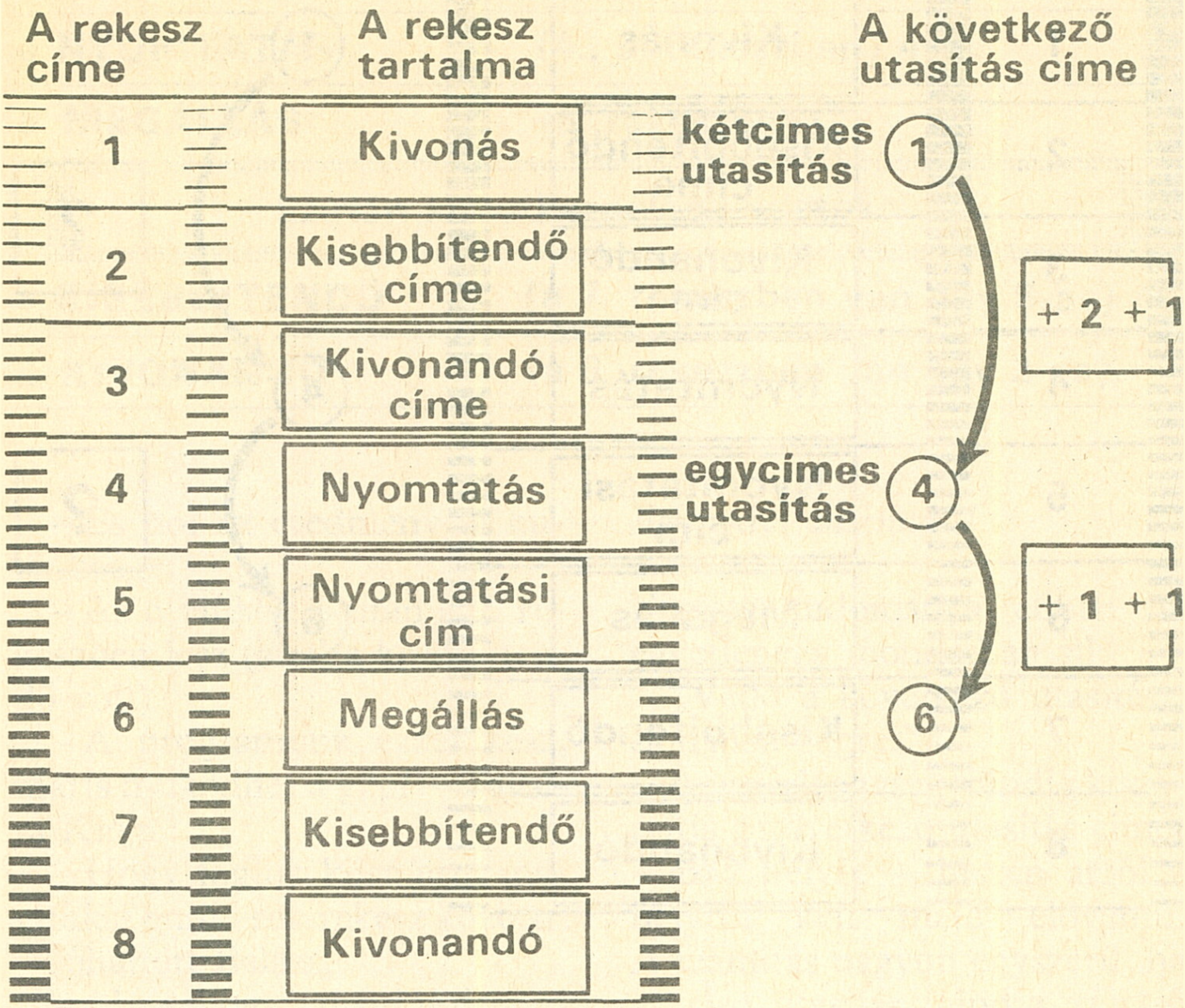
A vezérlőegység mindig kiszámítja a következő utasítás címét – azaz helyét –, hiszen arról a helyről kell majd elővenni. A számítás nagyon egyszerű, ha az első utasítás mindig az első rekeszben van.

Rögtön rájöttök a *címszámítás*

módjára, ha egymás alá rajzolom a rekeszeket:



– Megvan! Annyit kell hozzáadni, ahány címes az utasítás, és még egyet!



– Úgy van. Így lehet együtt tárolni a memóriában az utasításokat és az adatokat anélkül, hogy azok összekeverednének. Ez az ötlet, a *belső programtárolás* elve, Neumann Jánostól származik. Ma a legtöbb számítógép ezen az elven működik.

– Hogyan tudja kiszámolni a vezérlőegység a következő utasítás címét?

– Most jönnek a regiszterek! A vezérlőegységben az adatok rövid ideig való tárolására külön regiszterek szolgálnak. A legfontosabb közülük a *programszámláló,* mert ez tartalmazza a következő utasítás címét. A tartalma a munka megkezdésekor mindig 1, ezért kezdődik a program végrehajtása mindig az első rekesz tartalmának kiolvasásával.

– Ezt nem értem, apa!

– Tudod, hogy a vezérlőegységnek egyik része vezérlőjeleket állít elő az egész számítógép számára.

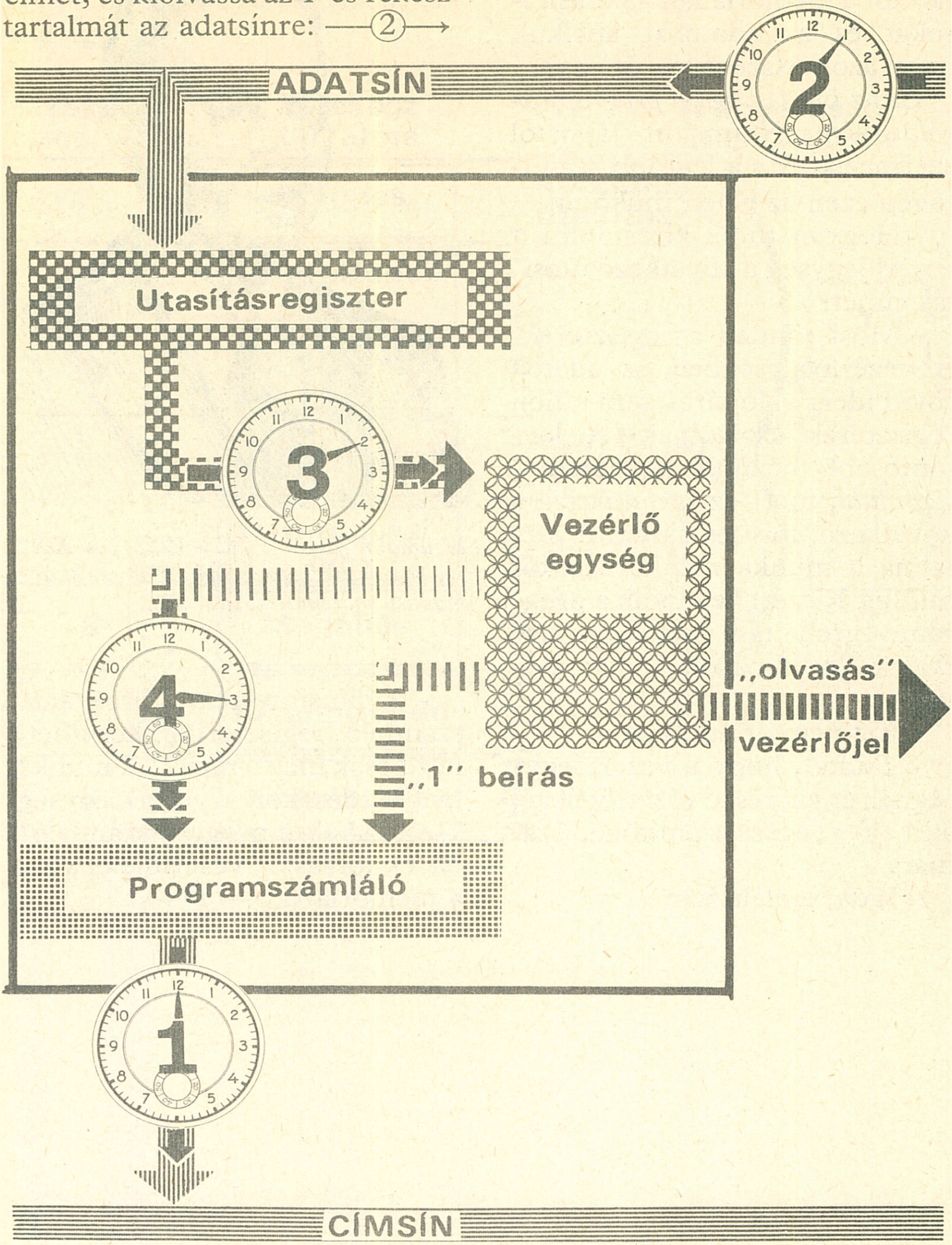
– Igen, emlékszem.



Neumann János (1903–1957), a XX. század egyik legnagyobb, magyar származású matematikusa

– A címsínre cím a programszámláló regiszterből kerülhet, nézd csak meg a rajzot! A munka megkezdésekor a vezérlőegység 1-est ír bele a programszámlálóba, és „olvasás" vezérlőjelet küld a memóriába: —①→

– Aha! A memória felismeri a címét, és kiolvassa az 1-es rekesz tartalmát az adatsínre: —②→



– Igen. Az adatsínről az első rekesz tartalma az utasításregiszterbe kerül, majd a vezérlőegységbe: —③→. A vezérlőegység a műveleti kódról felismeri, hogy hány címes utasítással van dolga, így annyit ad hozzá a programszámláló tartalmához, amennyit kell.

– Most a programszámlálóban 1 van, akkor ha 1-et ad hozzá, ki lehet olvasni a kisebbítendő címét: —④→ és így tovább!

– Pontosan ez történik! „Vedd a következő utasítást, és hajtsd végre!” – ez a mi mottónk.

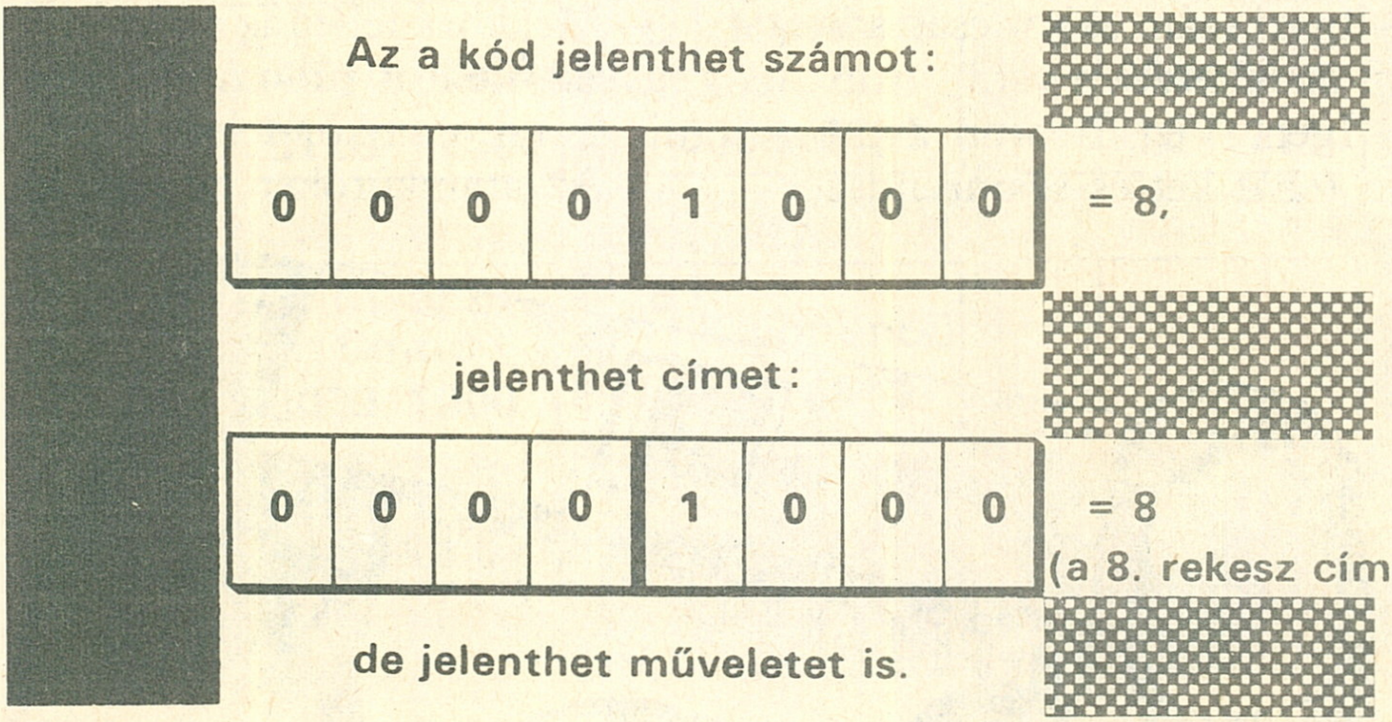
– Elég már a regiszterekből, apu! Inkább azt mondd meg, hogyan van a rekeszekben benne a kivonás meg a... a...

– Szóval a műveleti kódok! – fejezte be Kinga Balázs helyett a mondatot.

– Hát mi lehet egy rekeszben?

– Azt hiszem, egy rekeszben vagy egy szám, vagy valamilyen kód lehet – válaszolta Csaba.

– Pontosabban, mindig kód van.



Kódja van minden utasításnak, így a kivonásnak, a nyomtatásnak és a megállásnak is. Például ez is lehet:

A művelet	A kódja	Kiírom a kódokat részletesen is
KIVONÁS	5B	0101 1011
NYOMTATÁS	6C	0110 1100
MEGÁLLÁS	6F	0110 1111

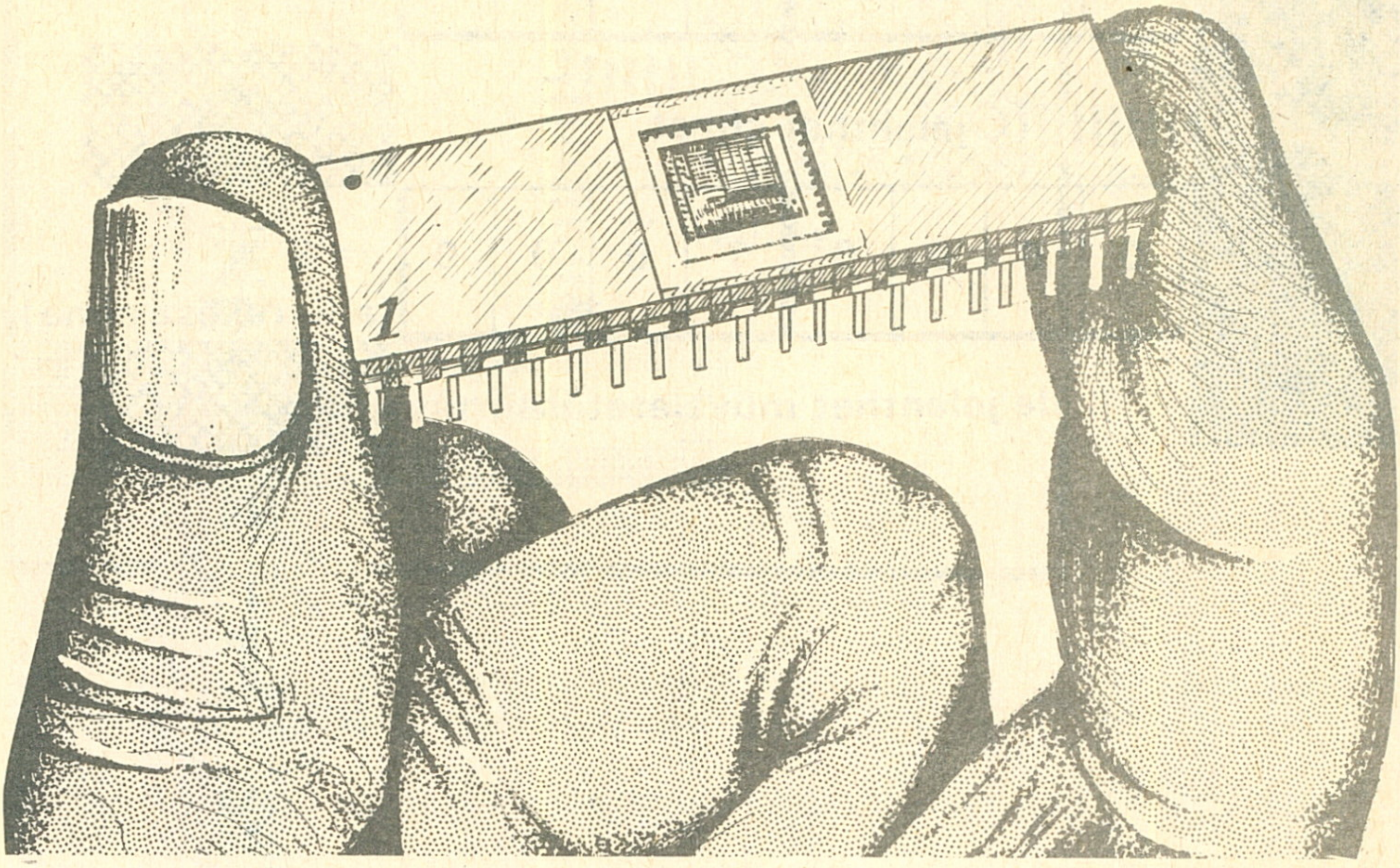
Ezek a kódok vannak az 1-es 4-es és 6-os rekeszekben, a többiben pedig a megfelelő fiókok címei.

– Igazi szám akkor csak a 7-es és 8-as rekeszben van!

– Igen. Végső soron a program tehát kettes számrendszerbeli kódok sorozata, amelynek jelentését a memóriában elfoglalt helyük alapján lehet megállapítani.

– Mekkora lehet az a vezérlőegység, nem is tudom elképzelni, annyi áramkör lehet benne!

– Az áramköröket tartalmazó



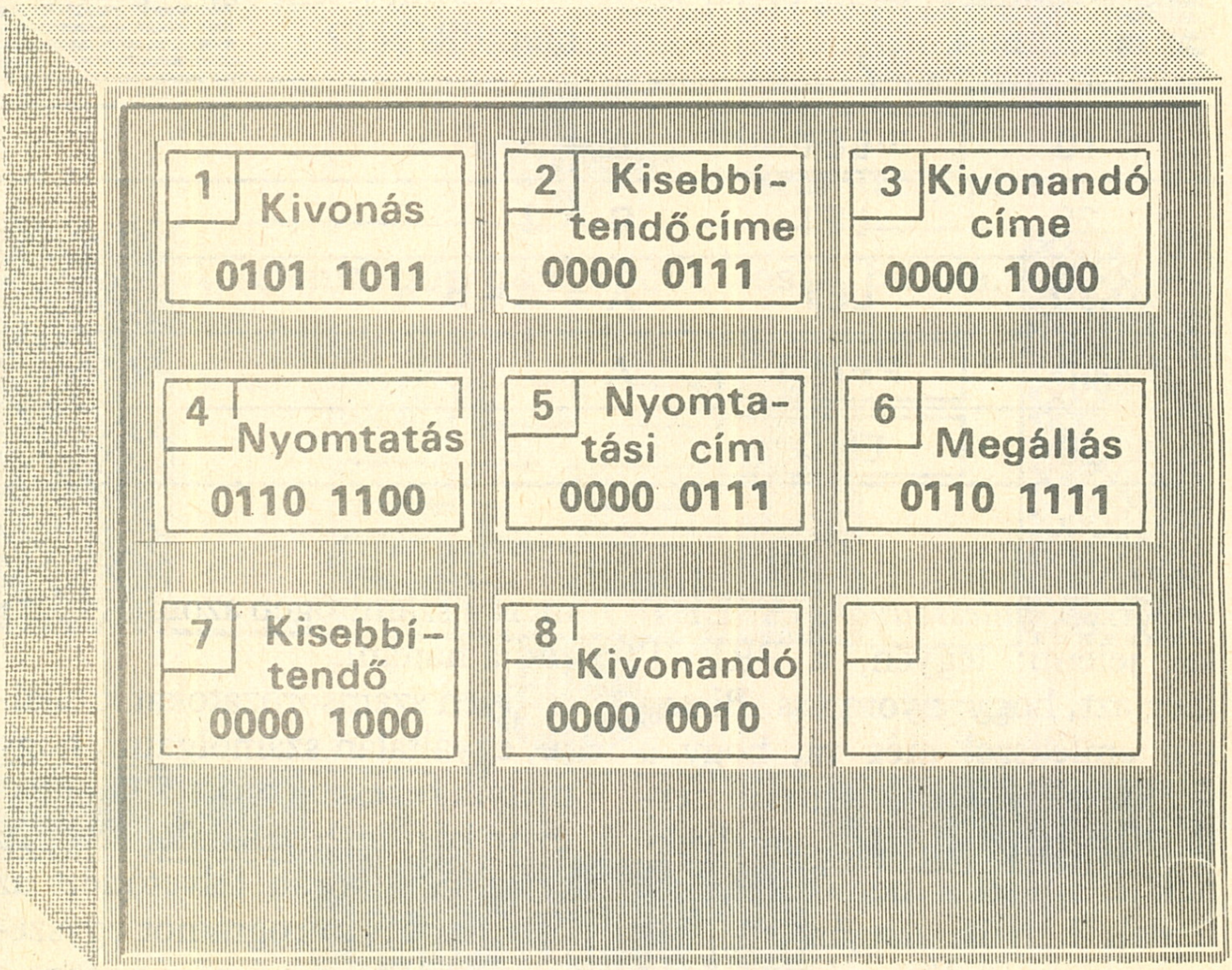
része akkora, mint a hüvelykujjadon a köröm, és még az egész feldolgozóegység is benne van!

– Hú, apu!

– Hú bizony! Egy olyan kis négyzeten, mint amit középen látsz, rajta van a teljes vezérlőegység és a feldolgozóegység. Az egésznek *mikroprocesszor* a neve.

– A rekeszekben csak kódok vannak? – törte meg az áhítatot Kinga.

– Igen. Lerajzolom neked még egyszer az egész programot,



és színessel ráírom minden egyes fiókra a tartalmát.

– Apu, úgy sajnállak téged! Nagyon nehéz lehet ezt a rengeteg számot kitalálni és beletölteni a fiókokba!

– Igazad van, elég nehéz dolog így programozni, de azért nem kell sajnálnod! Nem a *programozó* találja ki azt a sok számot, hanem – nem fogjátok elhinni – egy másik PROGRAM.

– Ezt is a számítógép csinálja?

– Ezt is.

Már nem sajnál senki

Nézzétek, itt van ugyanaz a program, amiről eddig olyan sokat beszéltünk:

– Ennyi az egész? – hüledezett Kinga.

– Ennyi.

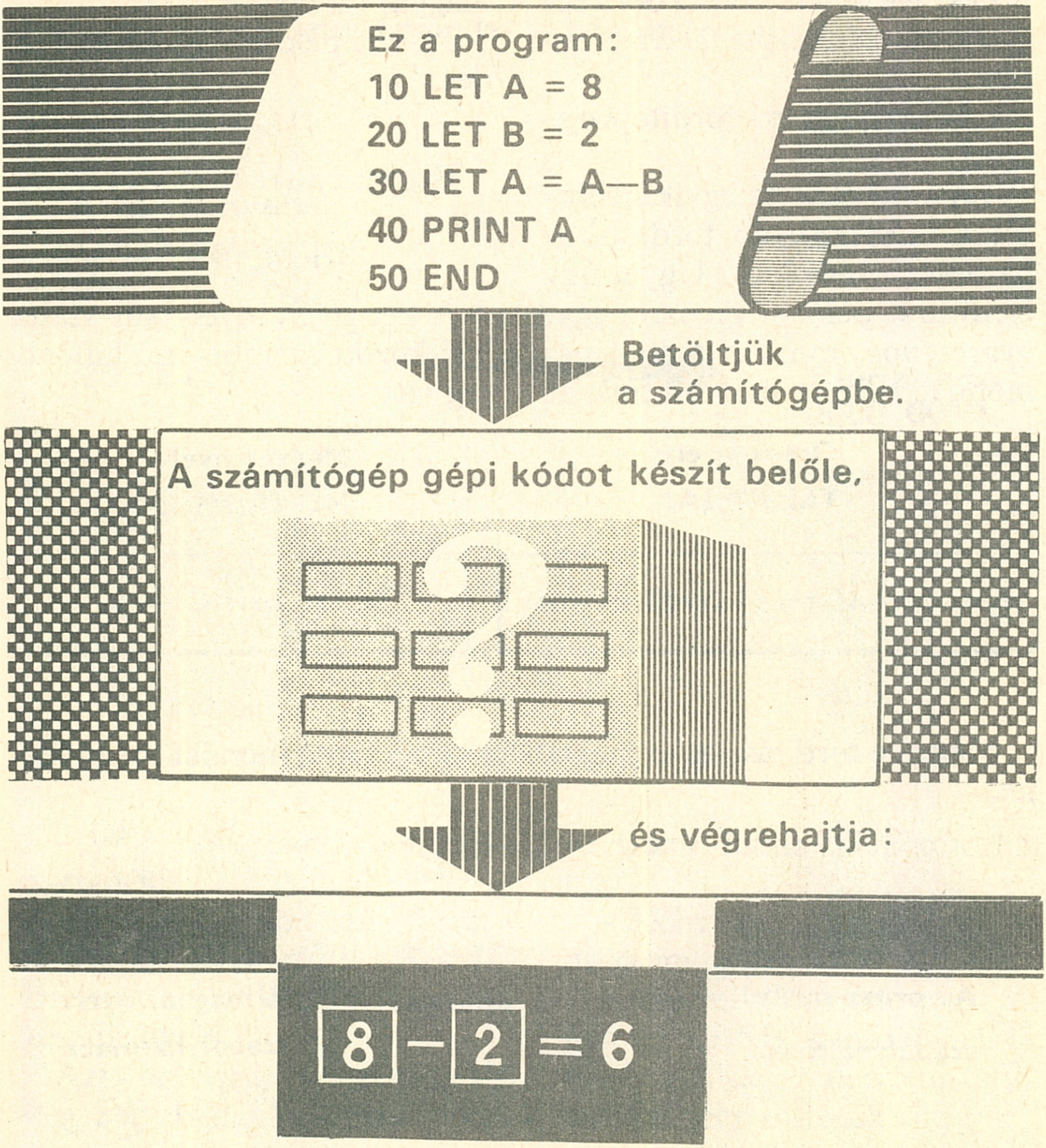
– És mit jelent az, hogy LET meg PRINT?

10	LET	A = 8
20	LET	B = 2
30	LET	A = A—B
40	PRINT	A
50	END	

– Ezek *kulcsszavak*. A LET azt jelenti: legyen, a PRINT meg azt, hogy: nyomtass. Biztosan emlékszel még rá, hogy a program kettes számrendszerbeli számok sorozata, amelyeknek jelentését a memóriában elfoglalt helyük alapján állapítja meg a vezérlőegység. Ezt a számsorozatot *gépi kód*nak vagy *gépi kódú program*nak nevezik, mert a számítógép azonnal végre tudja hajtani.

Ilyen számsorozatokat kitalálni, egyáltalán számolgatni, hogy melyik rekeszbe mi kerüljön, nagyon körülményes dolog. Éppen ezért már korán felvetődött az ötlet, hogy a programot egyszerűen le lehessen írni, és maga a számítógép írja meg ebből a leírásból a gépi kódú programot.

Esetünkben ez így van:



– Hogyan készíti a számítógép a gépi kódot? – kérdezte Balázs.

– Azt mondtad, hogy a számítógép olyan automata, amelyik csak végrehajtja a memóriájából előráncigált utasításokat.

– Ez változatlanul igaz. Most azonban a memóriájában egy

olyan program van, amely a működése során a leírásunkból gépi kódot készít. A neve *fordítóprogram*.

– Miért éppen fordítóprogram?

– Azért, mert az egyik nyelvről a másikra való fordításhoz hasonlóan működik. Ugye, úgy fordítunk például oroszról magyarra, hogy minden egyes orosz mondat helyett összeállítjuk az orosz mondat értelmét kifejező magyar mondatot.

Itt van például egy orosz kérdés:

ты куда?

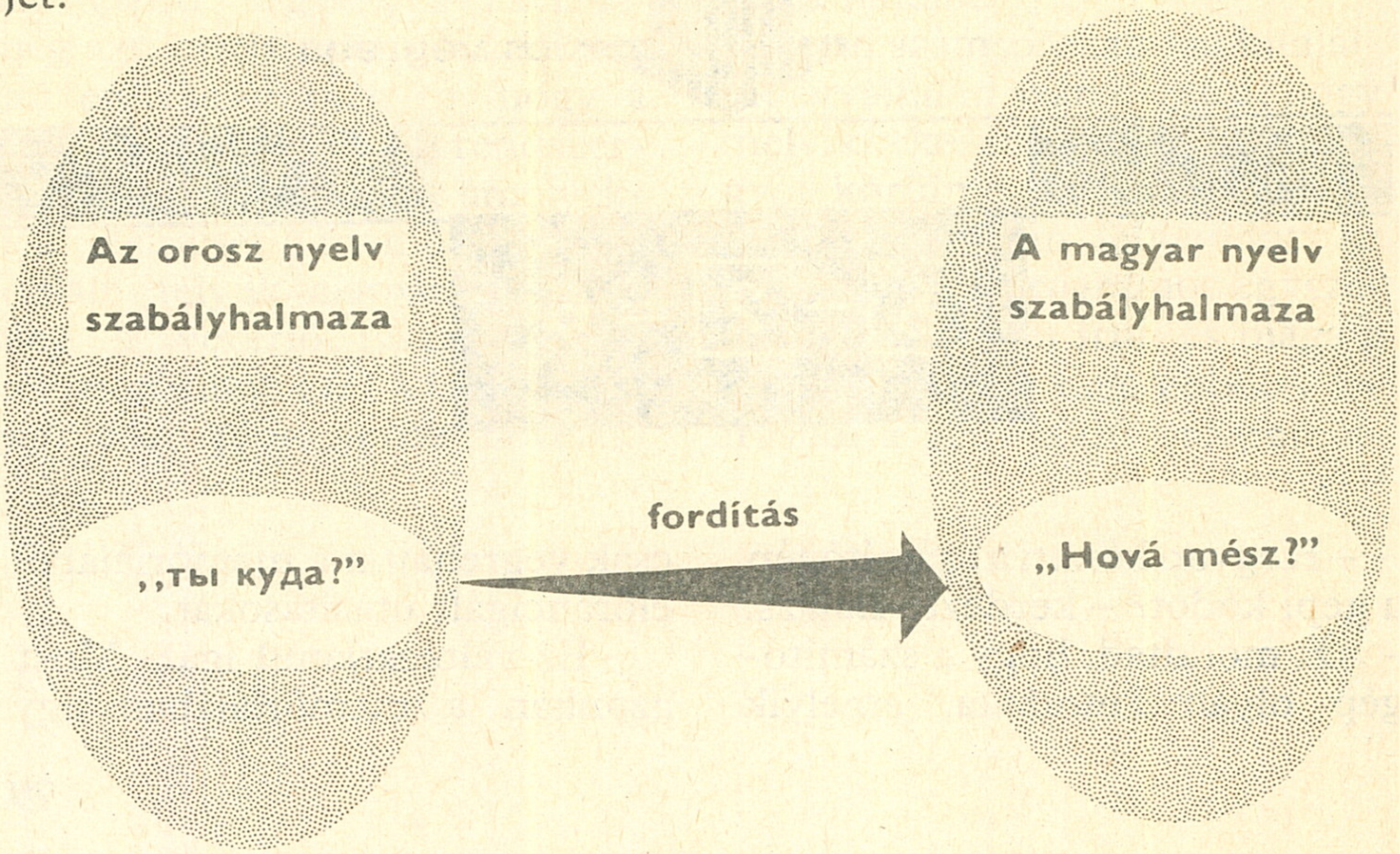
Ezt magyarra így kell fordítani:

Hová mész?

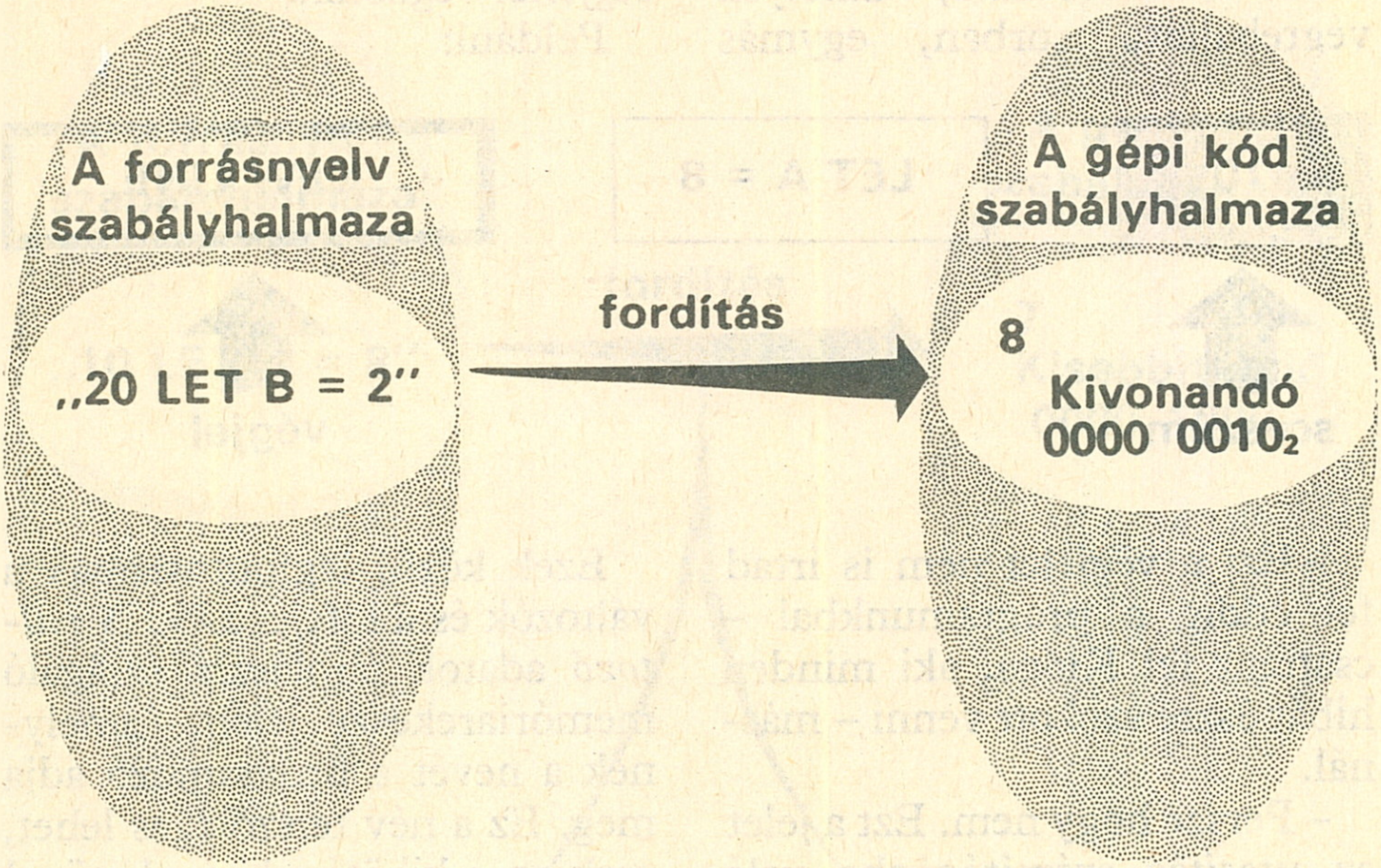
A két nyelv nyelvtani szabályai között szembetűnő különbség van:

Orosz nyelv	Magyar nyelv
ТЫ КУДА?	**HOVÁ MÉSZ?**
személyes névmás+kérdő névmás	kérdő névmás+ige

Mégis, a fordítás során megtaláljátok az orosz mondat megfelelőjét:



Így dolgozik a fordítóprogram is. Az egyik nyelven – ez a *forrásnyelv* – készült leírást mondatonként lefordítja gépi kódra, és a gépi kódot a memóriarekeszekbe tölti.



– Mi a neve ennek a forrásnyelvnek?

– *BASIC*. Több száz forrásnyelv van. Ezek közül a BASIC kifejezetten a kezdők számára készült nyelv.

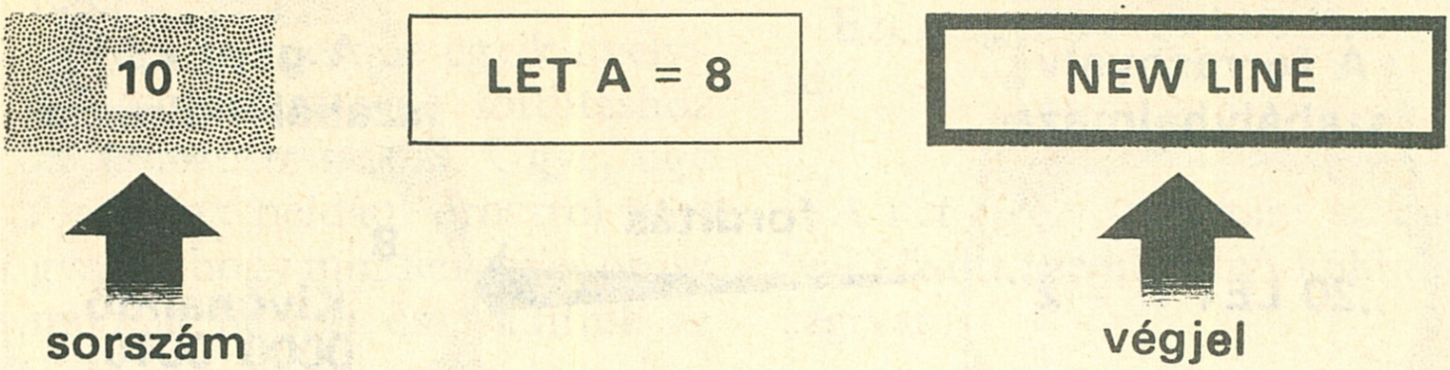
Azt ajánlom, ismerkedjetek meg a nyelv néhány szabályával, és akkor máris programozhattok!

– Akármilyen programot tudunk majd írni?

– Akármilyent nem, mert ahhoz egy BASIC-tanfolyam is kevés lenne, de az alapvető tudnivalókat meg fogjátok ismerni.

A BASIC program *forrásnyelvi utasítások* sorozata, amelyek végrehajtása sorban, egymás után történik. Egy utasítást egy sorba érdemes írni. Minden utasítás sorszámmal kezdődik, és *végjel*lel végződik.

Például:



– Ezt a végjelet nem is írtad fel eddig a programunkba! – csattant fel Kinga, aki minden hibát észre szokott venni – másnál.

– Persze hogy nem. Ezt a jelet az utasítás számítógépbe való bevitelekor egy billentyű lenyomásával kell előállítani. A billentyű neve „NEW LINE”, azaz „új sor”. A következőkben nem fogom az utasítások után feltüntetni, gondoljátok oda.

Szóval a sorszám után jön egy kulcsszó, utána pedig a *nyelv* egyéb *elemei* következhetnek.

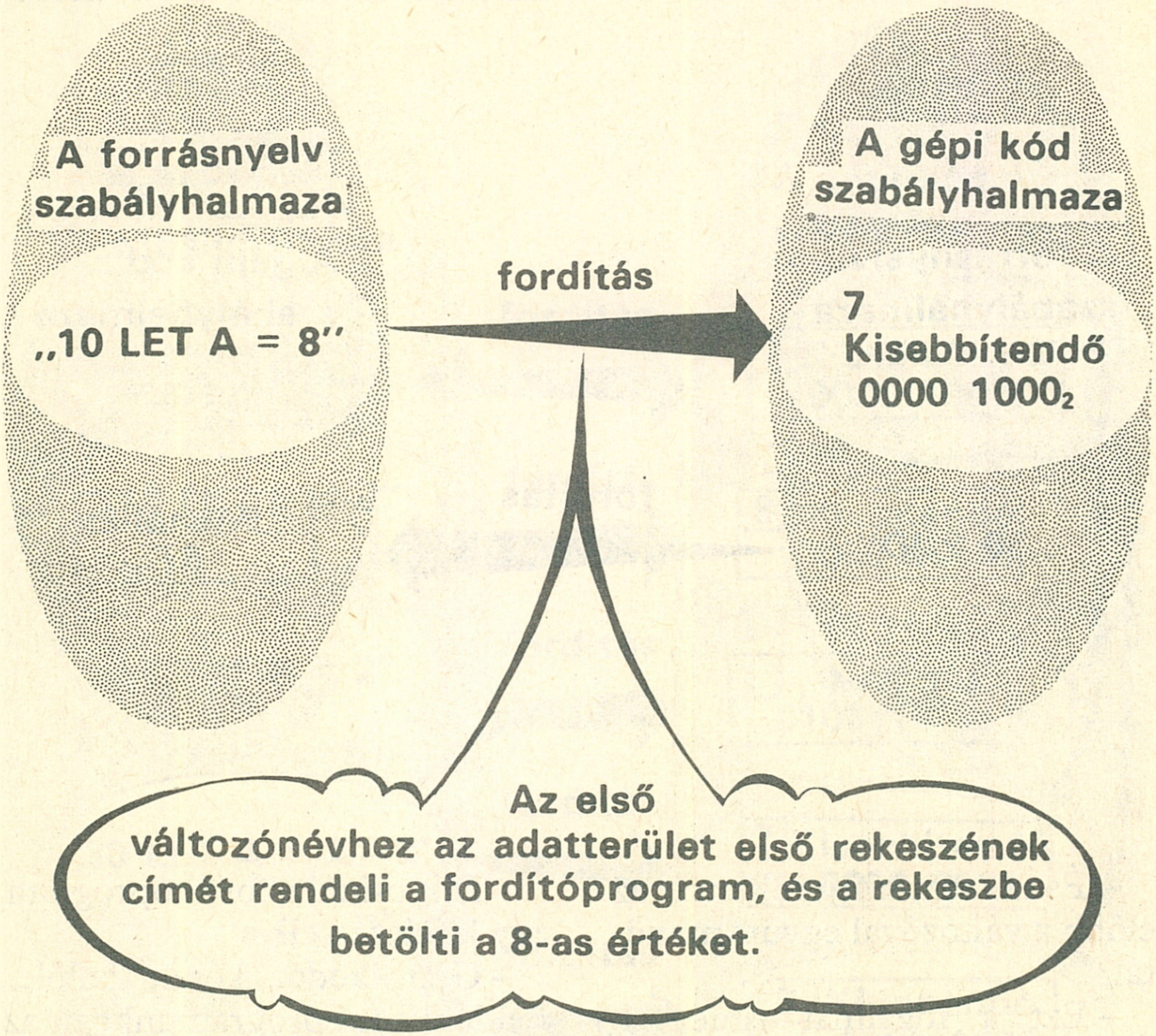
Ezek közül legfontosabbak a változók és a kifejezések. A *változó* adatok tárolására szolgáló memóriarekeszt jelent, amelynek a nevét a programozó adja meg. Ez a név tetszőleges lehet, csak az a kikötés, hogy betűvel kezdődjön.

Ilyen *változónév* volt példánkban az A és a B.

– Mire jók ezek a nevek? – kérdezte Kinga, mert egészen idáig nem figyelt.

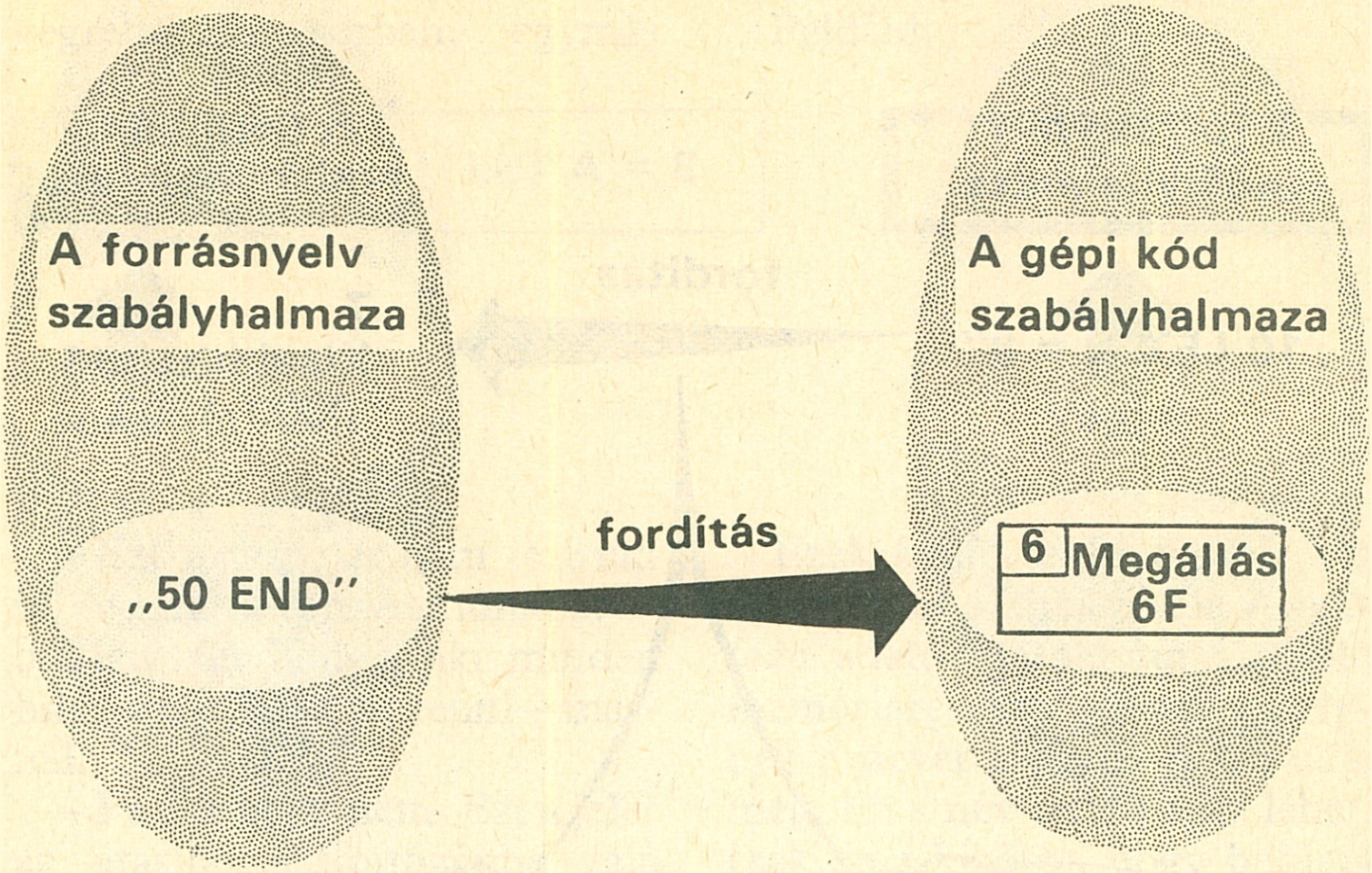
– Arra, hogy ne kelljen a memóriacímeket számmal kiírnod. A fordítóprogram minden egyes

változónévhez hozzárendel egy memóriarekeszt az adatterületen, és annak a címét beírja a gépi kódú utasításba.



– Akkor a B változó kerül a 8-as rekeszbe?

– Igen. A forráslistában ez a második változónév, ehhez a fordítóprogram az adatterület második rekeszét rendeli hozzá!



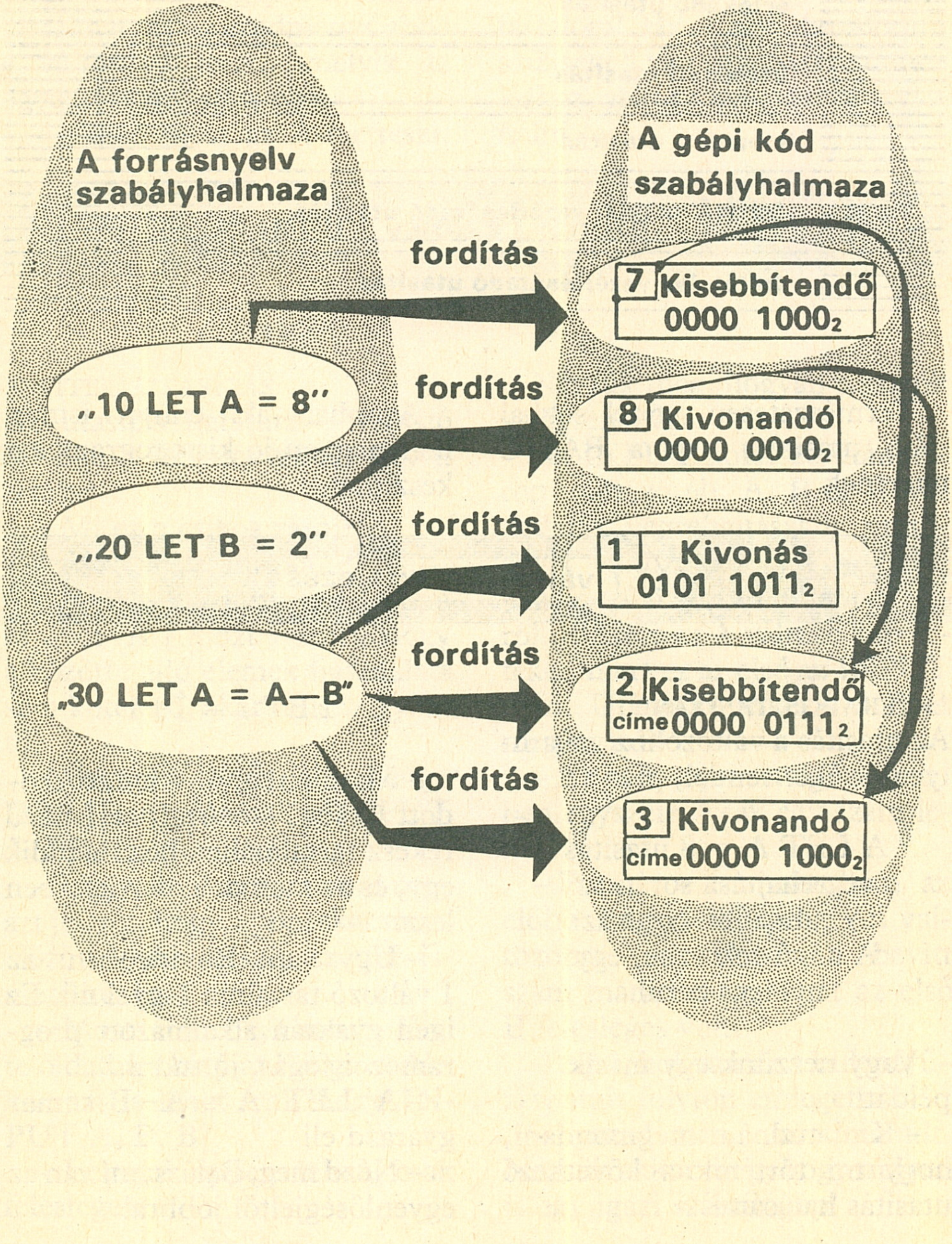
– És mi az a kifejezés, amit az előbb a változóval együtt mondtál?

– Ezt a fogalmat ismeritek már a matematikából: a *kifejezés* a változók közötti kapcsolat leírása. Ilyen kifejezés például az Y = 2X+3 vagy az A = A–B. Itt az Y, X, A és B a változók.

– Mit csinál a fordítóprogram egy kifejezéssel?

– Gépi kódot készít belőle. Például a mi programunkban az A = A–B egy kifejezés. Ez a harmadik sorban lévő utasítás. Mire a fordítóprogram a fordítás során ide eljut, már lefoglalt egy-egy rekeszt az A és B válto-

zóknak, így a kivonás gépi kódú utasításában a címeket is ki tudja tölteni.



Az öt legfontosabb utasítás a következő:

értékadó utasítás

bemeneti utasítás

kimeneti utasítás

feltétel nélküli vezérlésátadó utasítás

feltételes vezérlésátadó utasítás

Természetesen, ennél sokkal több utasítása van a BASIC nyelvnek. Azonban aki ennyit ismer, már nagyon jó kis programokat készíthet.

LET változó = kifejezés

ÉRTÉKADÓ UTASÍTÁS

Az utasítás a változóhoz valamilyen értéket rendel, pl.:

A LET A = 8 utasítás
végrehajtása során az
A változóba (azaz az
A nevű rekeszbe) egy
nyolcas kerül.

Vagy nézzünk egy másik példát!

– Ki tudná megmondani, hogy mi történik a következő utasítás hatására?

LET İ = İ+1

– Majd én, apu! – nyújtózkodott Csaba. – Szóval, az İ nevű rekesz tartalmához hozzáadódik egy, és az megint az İ rekeszben lesz.

– Úgy van. Rövidebben: az İ változó tartalma 1-gyel nő. Ez igen gyakran alkalmazott programozói fogás, jó tudni.

– A LET A = A – B-t magyarázd el!

– Nézd meg, Balázs, mi van az egyenlőségjeltől jobbra!

– Az A – B.

– Ez az. Az A változó tartalmából kivonja a gép a B változó tartalmát, és az eredményt abba a változóba teszi, amelyik az egyenlőségjel előtt van.

– Akkor most az A-ba teszi, de ezzel elrontja azt, ami eddig benne volt!

– Úgy van, volt = nincs régi tartalom! Ha a régi tartalomra szükséged van egy másik számoláshoz, akkor ilyet nem illik csinálni.

INPUT változó ⟦,változó...⟧

BEVITELI UTASÍTÁS

Az utasítás hatására a változó felveszi a billentyűzetről bevitt értéket.

– Mi ez a csíkos zárójel?

– És az a három pont?

– Jelölések, most gondoltam ki őket. A ⟦ ⟧ azt jelenti, hogy a közötte álló elemek használata sem kötelező, akár el is hagyható.

– Vagyis az is jó, hogy INPUT A, és ez is: INPUT A, B?

– Úgy van. A három pont meg azt jelenti, hogy a pontokat megelőző elemek tetszés szerint ismételhetők.

– Most én jövök, Csabi! – lökte oldalba a bátyját Kinga. – Akkor azt is lehet írni, hogy INPUT A, B, B?

– Lehetni éppen lehet, csak felesleges.

Nézzük meg, hogyan történik a végrehajtása.

Az INPUT kulcsszó után három változó áll: A, B és B. Az utasítás végrehajtásakor mindegyik változóba egy-egy érték, vagyis szám kerül a billentyűzetről. Először az A változó kap értéket, mert ez az első az INPUT után. Utána következik a B és megint a B.

– Ez nem jó, mert egy rekeszben egyszerre csak egy szám lehet!

– Persze, ezért mondtam az előbb, hogy felesleges. A B változóban az utolsónak bevitt szám marad meg, tehát az első B felesleges volt.

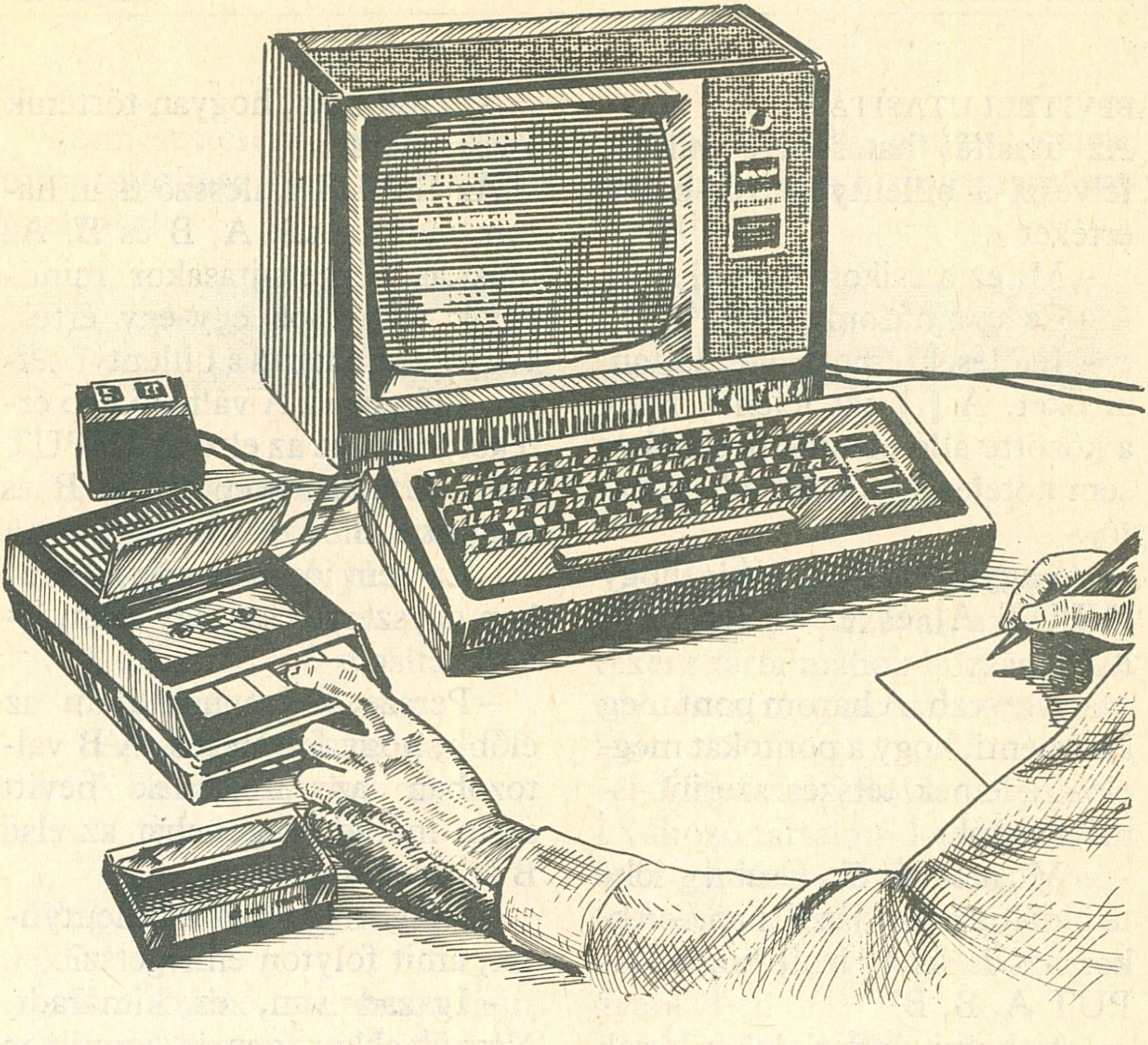
– Apu, hol van az a billentyűzet, amit folyton emlegetsz?

– Igazad van, ez kimaradt. Nézzük akkor meg, hogy milyen is egy igazi számítógép!

Kisgépek – nagygépek

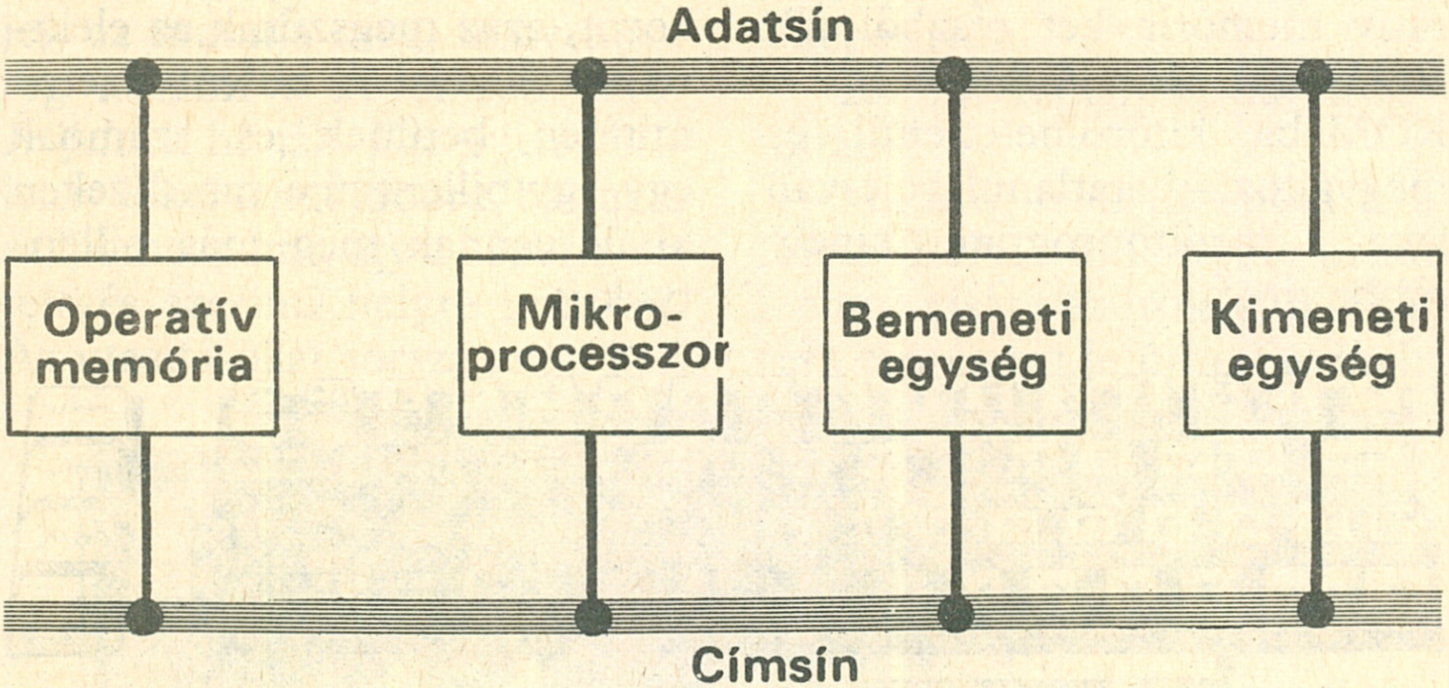
Ma már egyre több iskolában láthattok számítógépet, talán a tiétekben is van. A legegyszerűbbet a képen láthatod.

Billentyűzet, egy tápegység, egy közönséges kazettás magnetofon, no meg egy tv-készülék, a szükséges vezetékekkel összekötve. A tv színes is lehet, ha a gép is tud színeket előállítani.



– Az a sok vezeték az adatsín?

– Majdnem. Ezek többnyire egyetlen mikroprocesszort tartalmazó gépek, háziszámítógépeknek, iskola-számítógépeknek is nevezik őket.



– Hol van a feldolgozóegység? – rémlett fel valami Kingának.

– A mikroprocesszorban! Jó reggelt! Most nézzétek meg a billentyűzet dobozát. Ebben a dobozban van benne a mikroprocesszor, az operatív memória, a címsín, az adatsín és a bemeneti egység is, amelyhez a billentyűzet tartozik.

– A kimeneti egység a tv?

– Igen. Sok háziszámítógép egyik kimeneti egysége a televíziókészülék, amely lehet színes is. Ilyen például a ZX81, a Spectrum, a VC–20, vagy a Commodore–64.

– Miért kell hozzá a magnetofon?

– Nem kell feltétlenül, de jó, ha van *háttértár*unk. Ha ugyanis kihúzod a konnektorból a számítógép csatlakozóját, akkor az operatív memória rekeszei mind elvesztik a tartalmukat, bármi is volt bennük.

– Volt program, nincs program!

– Úgy van. Ha van háttértárad, ami itt most éppen egy kazettás magnetofon, akkor a memória tartalmát kiírathatod a kazettára, és azon megmarad a programod.

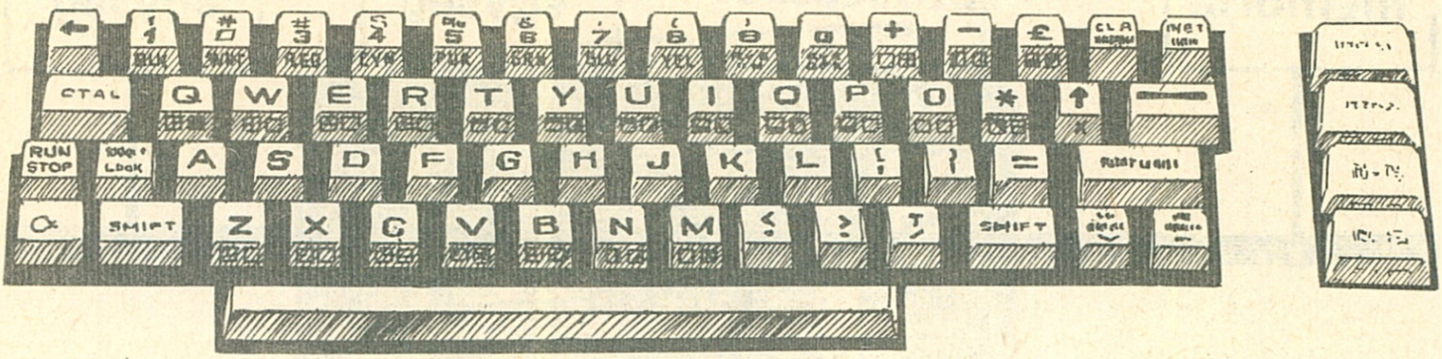
Sőt más által írt programot is tudsz használni, mert a kazettáról a memóriába is lehet betölteni programot.

– Ha kihúzom a hálózati csatlakozót, mi történik a fordítóprogrammal?

– Szerencsére semmi. Az operatív memória két részből áll, *ROM*-ból és *RAM*-ból. A ROM-ba kitörölhetetlenül és megváltoztathatatlanul bele van írva a fordítóprogram, amely többnyire a…

– BASIC!

– A RAM az a memóriarész, amelyet a programozó használ: ez törlődik, ha kihúzod a csatlakozót, azaz megszűnik az elektromos ellátás! A billentyűzeten minden betűnek és számnak egy-egy billentyűje van. Ezeken kívül vannak még más billentyűk is.

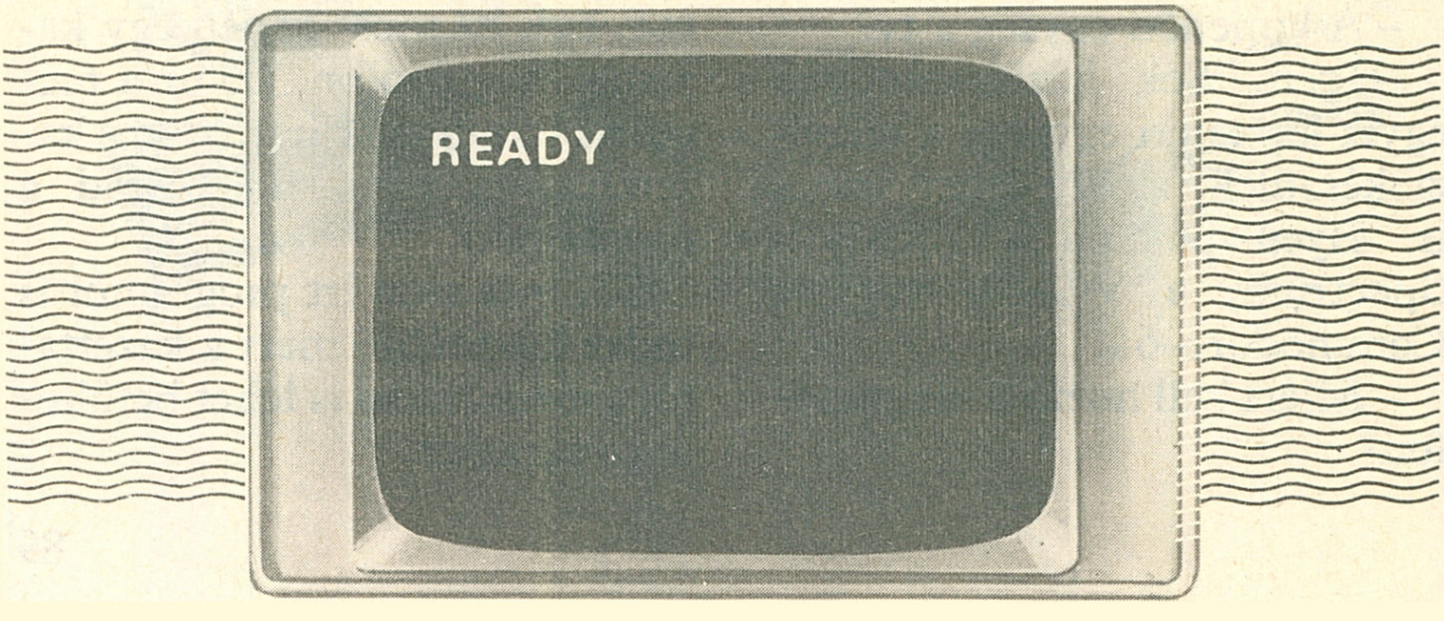


– Ezek között nincs is NEW LINE! – kereste Balázs a billentyűt, mert nagyon szeret az írógéppel játszani.

– Vannak olyan gépek, mint például ez is, amelyeken a RETURN billentyű állítja elő a végjelet. Tehát a RETURN és a NEW LINE billentyűket azonos értelemben használjuk.

– Taníts meg rá, apu!

– Amikor bekapcsolod a számítógépet, a képernyőn megjelenik ez a felirat: *READY*. Azt jelenti, hogy a számítógép kész a programotok fogadására.

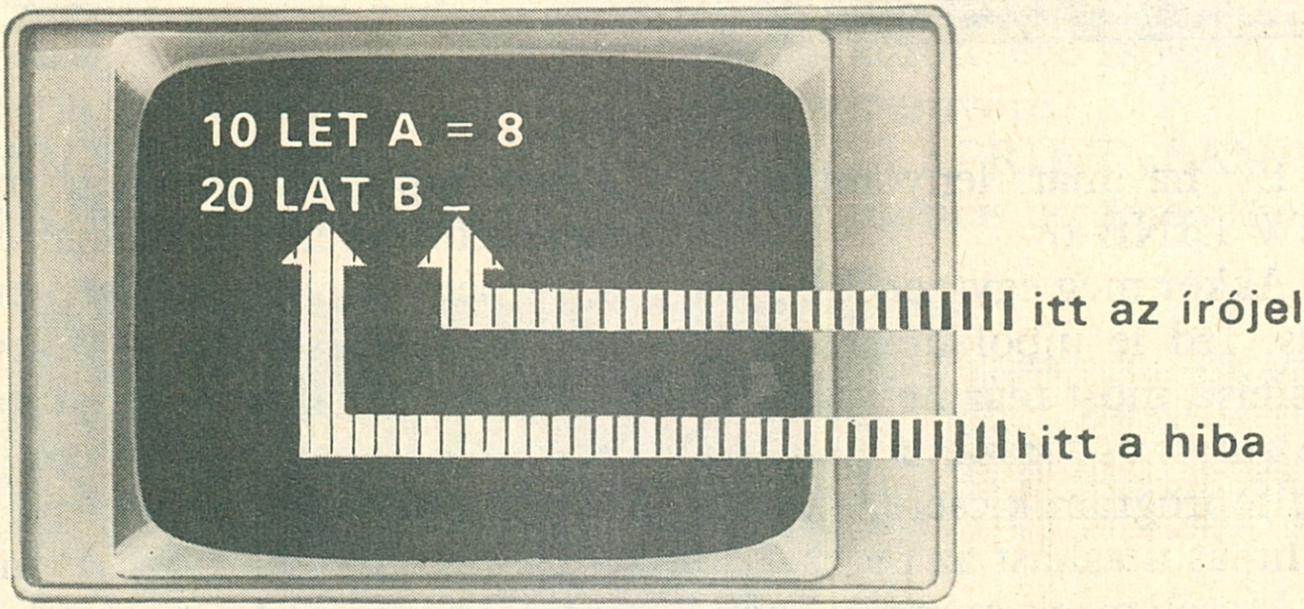


Ezután ugyanúgy, mint amikor az írógépen írtok, a billentyűzeten lekopogjátok az utasításokat. A képernyőn van egy kis jel, ez mutatja meg, hogy hová fog kerülni a következő betű. *Írójel*nek hívják. Ezt a jelet billentyűkkel lehet jobbra, balra, le és föl mozgatni, így a képernyőn tetszés szerinti helyre írhattok. Az utasításokat sorszámmal kell kezdeni, és a NEW LINE billentyűvel befejezni.

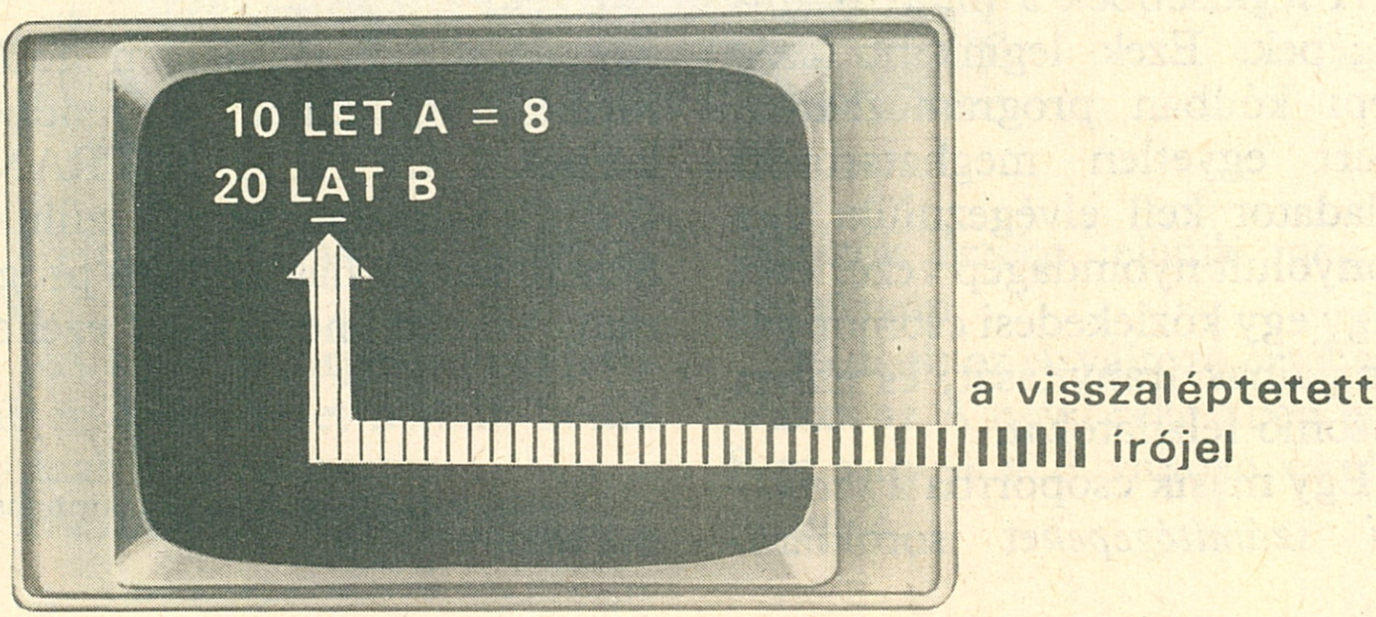
A lenyomott billentyűnek megfelelő jel azonnal megjelenik a képernyőn.

– És mit csináljak, ha elrontottam?

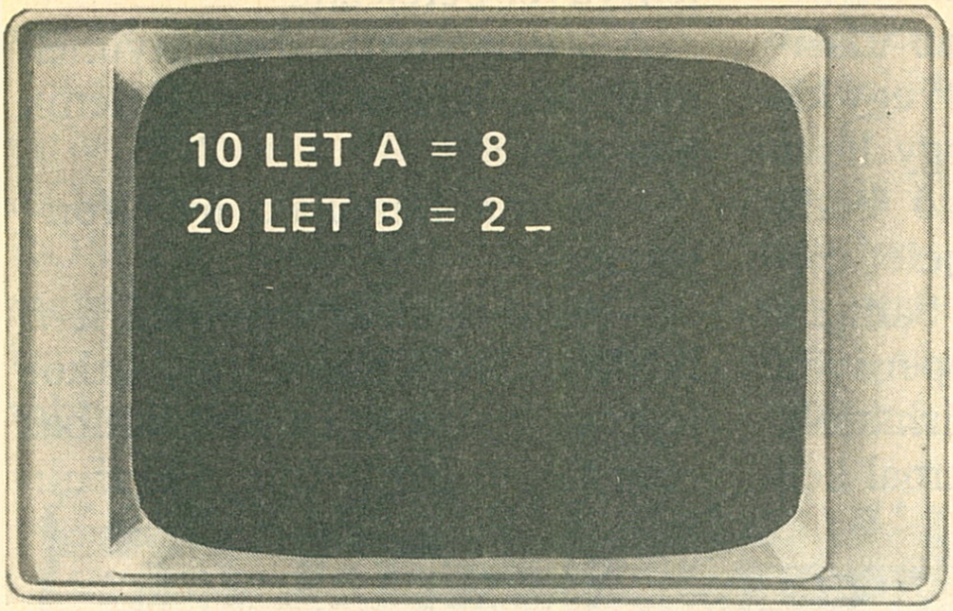
– Ha még nem nyomtad le a NEW LINE billentyűt, akkor léptesd vissza az írójelet a hibás rész elé, és írd le újra. Például:

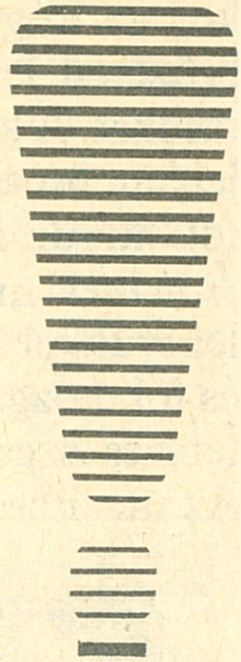


Visszalépteted az írójelet:



és leírod újból, amit akarsz.





– És ha már lenyomtam a NEW LINE-t?

– Akkor még egyszerűbb a javítás. Írd le újból ugyanazt az utasítást, most már helyesen.

Az utasítás sorszáma alapján a fordítóprogram kicseréli az előző, hibás utasítást az újabbal.

– Mesélj még a számítógépekről, apu!

– Ó, nagyon sokféle van ám! A legkisebbek a mikroszámítógépek. Ezek leginkább csak gépi kódban programozhatók, mert egyetlen meghatározott feladatot kell elvégezniük: egy bonyolult nyomdagép vezérlését vagy egy közlekedési csomópont lámpáinak irányítását, és ehhez hasonló feladatokat.

Egy másik csoportba a *személyi számítógépek*et sorolnám. Azért kapták a nevüket, mert egyetlen ember munkájának elősegítésére valók, s emellett, mint egy személyes tárgy, könnyen szállíthatók.* Ezeknek már jóval nagyobb memóriájuk van, és háttértárként mágneslemezes egységeket használnak. Mágneslemezen jóval több adatot lehet tárolni (több száz kbyte-ot vagy akár több *Mbyte*-ot is), mint a kazettán.

Nemcsak egy, hanem több fordítóprogrammal is rendelkeznek (PASCAL, FORTRAN, C stb.), választani lehet közülük. Számtalan jól használható program készült hozzájuk: ezeket

* Ilyen gépet láthattok a hátsó borító bal alsó sarkában

gyűjtőnéven *software*-nek hívják.

És egy nagyon fontos dolog:

vezetéken csatlakoztathatók egy még nagyobb teljesítményű számítógéphez.

1 **PRINT „szöveg"**

2 **PRINT változó [[, változó. . .]]**

A maradék BASIC utasítások

KIVITELI UTASÍTÁS

1. *PRINT „szöveg",*
2. *PRINT változó [[változó]],*

Az 1. utasítás hatására az idézőjelek között lévő szöveg kiíródik a képernyőre.

Például: a PRINT „SANDOKAN" utasítás eredménye ez:



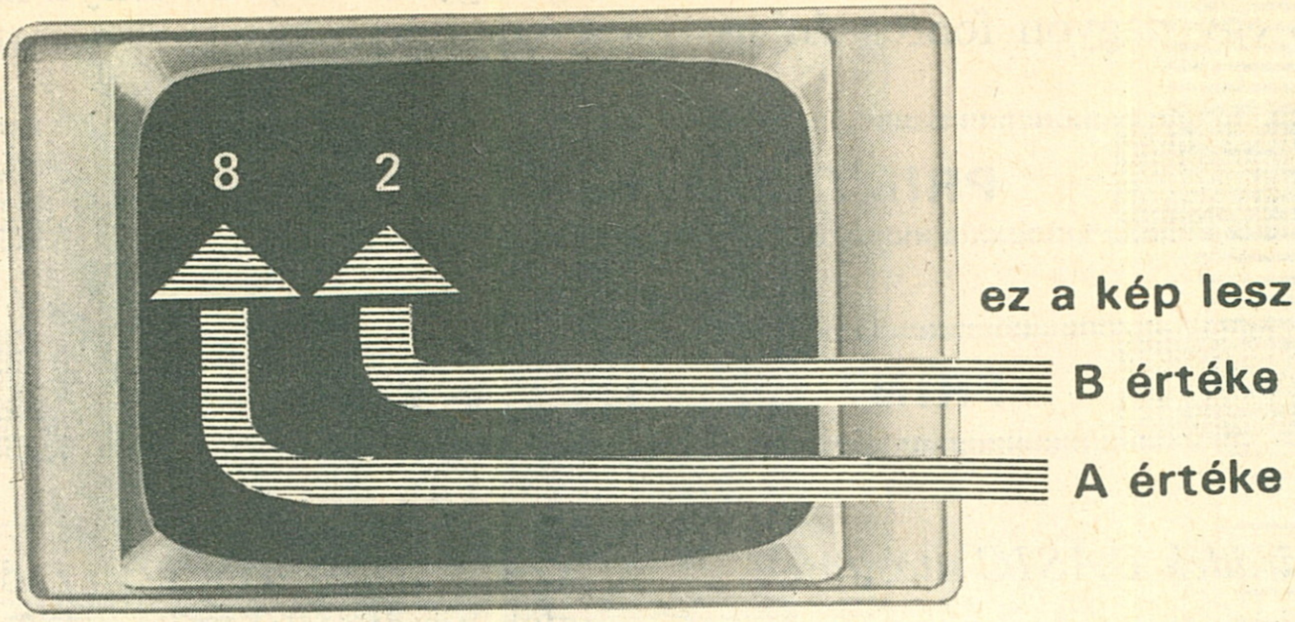
– Hol maradtak az ékezetek, apu?

– A legtöbb forgalomban levő gépen csak ékezet nélküli betűk vannak – egyelőre.

A 2. utasítás végrehajtásakor a változók értéke jelenik meg a képernyőn.

Például: az A változó értéke 8 – azaz az A nevű rekeszben egy 8-as van –, a B változó értéke pedig 2.

Ekkor a PRINT A, B utasítás eredménye:



FELTÉTEL NÉLKÜLI VEZÉRLÉSÁTADÓ UTASÍTÁS

Az utasítás hatására a program végrehajtása a megadott sorszámú utasításon folytatódik.

Például: itt van ez a programrészlet

```
1ø LET A=8
2ø LET A=2
3ø GOTO 5ø
4ø PRINT „MIT AKARSZ?"
5ø LET C = A–B
```

Mivel az utasítások végrehajtása egymás után, sorban történik, ezért ez lesz a sorrend:

10-es, 20-as, 30-as, majd 50-es, mivel a GOTO utasításban lévő sorszámon folytatódott a végrehajtás.

– Ezért kell hát az utasítások elé a sorszám!

– Ezért is. Ha hallgattok rám, tizesével számozzátok a sorokat, úgy, mint én, mert a gyakorlatban ez nekem nagyon jól bevált.

GOTO sorszám

Működése:

IF állítás THEN utasítás

(az IF szó jelentése: ha, a THEN szóé: akkor)

– ha az állítás igaz, akkor az utasítás végrehajtódik;
– ha az állítás nem igaz (azaz HAMIS), akkor az utasítás nem hajtódik végre.

Például:

```
1. 1ø INPUT A
   2ø IF A=2 THEN
   GOTO 1ø
   3ø LET B = A+2
```

Ez a program beolvas egy számot az A változóba. Ha ez a szám 2, akkor újból a 10-es utasítás következik.

Ha ez a szám nem kettes, akkor az állítás hamis, ezért a program végrehajtása a soron következő, tehát a 30-as utasítással folytatódik.

Ezzel be is fejeztük a legfontosabb BASIC utasításokat.

– És mitől fog megállni a mikroprocesszor? – kérdezte Balázs vészjósló hangon.

– Igaz. Kell egy megállító utasítás is:

END

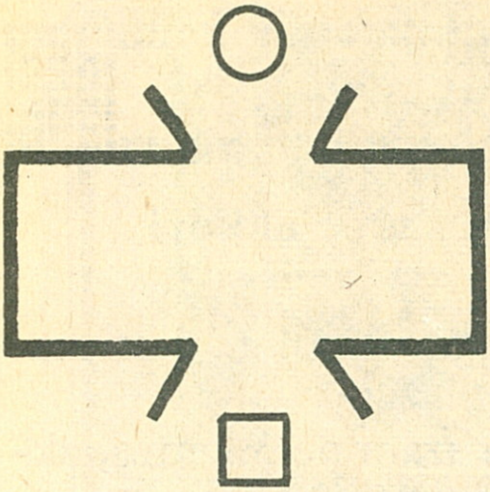
– Akkor én már írom is a programomat! – mondta Kinga. – 10 LET... – De mit is írjak?

– Jó vagy! A babaruhát is előbb varrod, és csak aztán szabod ki? – nevetett Csaba, és csak komoly erőfeszítéssel tudtam szétszedni az elszántan verekedő feleket.

– Álljatok meg! A programozásnak is éppúgy megvan a maga logikus sorrendje, mint bármely más céltudatos (emberi) cselekvésnek. Ezzel foglalkozunk majd a következő fejezetben. Az erőtöket meg ne egymáson, hanem ezen a három példán próbálgassátok addig is!

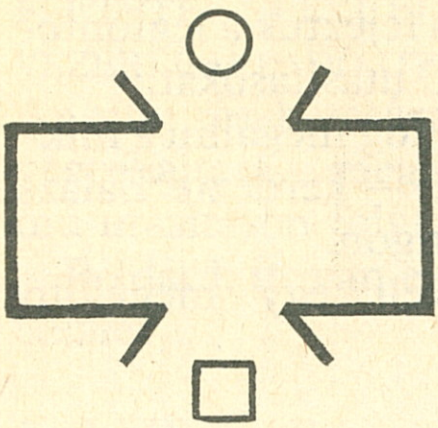
GONDOLKOZZATOK!

1. Készítsd el az alábbi gép programját:



szabály: □ = ○ + 4

2. Mi lesz a programja ennek a gépnek?



szabály: □ = ○ + 4

ha ○ = −2

a gép megáll.

3. Írasd ki PRINT utasításokkal a képernyőre a ○ és a □ értékét! (A gép szabálya ugyanaz, mint a 2. példában.)

– Apa, baj van a programoddal! – szólt a nagyfiam. – Megnéztem az első példa megoldását, és az iskolában kipróbáltam egy gépen. Jól is működött, de egyszer azt a számot adtam meg, hogy 1984,1984 és nem számolt pontosan. Voltak ott fiúk, s elindítottuk újra a programot. Most kisebb számokat adtam meg, megint jól működött, csak sohasem akart megállni.

– Igazad van, valóban hibás az a program. Én sem tartottam be azokat a szabályokat, amelyeket Kingának emlegettem.

A feladatmegoldás sorrendje

Mindössze négy sorban össze lehet foglalni a helyes sorrendet:

1. A feladat megértése.
2. Algoritmuskészítés.
3. Programozás.
4. A program kipróbálása.

Az első hiba oka mindjárt az első szabály megsértése volt. Ha ugyanis alaposan átgondoltam volna a példát, rájöttem volna arra, hogy a kör ○ bármilyen számot, akár egy tizenötjegyűt is jelenthet. Így akár nyolc vagy több számjegyűt is.

– Emlékeztek még, hogy mi a változó?

– „Adat tárolására szolgáló memóriarekesz” – idézett Kinga.

– Úgy van. Az adat pedig lehet szám vagy egy *szöveg*. A BASIC nyelvben számok tárolására háromféle *változótípus* is van:

– egész,
– egyszeres pontosságú és
– kétszeres pontosságú.

A változó típusát a nevében kell külön jellel jelezni.

Az *egész típusú változó* csak egész számok tárolására szolgál. Jelzése a százalékjel. Például az A% változó értéke lehet 0, –15, 28, sőt 32768 és –32768 között akármilyen egész szám, de 1,2 , vagyis tizedestört már nem lehet.

– Nem lehet 50 000 sem? – csodálkozott Balázs.

– Nem.

– Ha nagyobb számmal akarsz dolgozni, akkor az *egyszeres pontosságú változó*t kell használnod. Jelzése az, hogy nincs külön jelzése.

Az ilyen változók legfeljebb 6

számjegyből állhatnak, és tizedespontot is tartalmazhatnak.

Például a hibás programban a KO és a NE is ilyen változók voltak.

A KO lehet 999999, 0,15 vagy bármi más szám, de csak maximum hatjegyű.

– Akkor ezért kaptam eredményül 1988,19-et, mikor azt adtam meg, hogy 1984,1984!

– Igen. Mivel a KO egyszeres pontosságú változó, az utolsó két tizedesjegyet a fordítóprogram nem veszi figyelembe. Ha *duplapontosságú változó*t használtam volna – ennek a jele a # –, akkor maximum 16 számjegyből álló számot is megadhattál volna.

– Hát ezt kipróbálom!

– Próbáld ki! A KO helyett KO#-t a NE helyett NE#-t írj mindenhová, és akkor nem fog elromlani az 1984,1984-re sem.

– És a másik hibával mi lesz? – így a lányom.

– Azt is kijavítjuk, de előbb ismerd meg a szöveges változót, különben nem tudod kiíratni a képernyőre azt sem, hogy „KINGA KÁRÖRVENDŐ!”

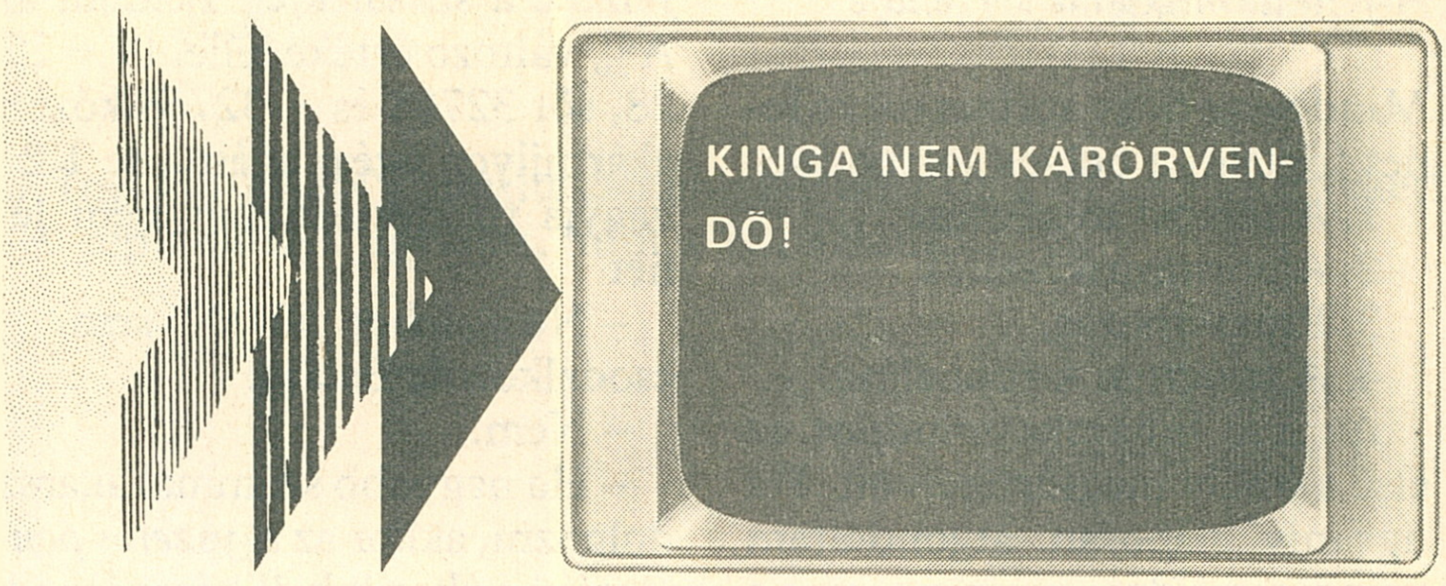
– Apu!

– Nos, a *szöveges változó* jele az $. Például az A3$ vagy a B$ változók ilyenek. Ezekbe maximum 255 jel hosszúságú szöveget írhattok idézőjelek között.

Ezek után így teljesíthető a kívánságod, Kinga:

```
1ø LET A$ = „KINGA
NEM KÁRÖRVENDŐ!”
2ø PRINT A$
3ø END
```

Ez lesz az eredménye:



Miért nem akart a program megállni?

– Ha alaposan átgondoltátok a feladatot, számba vettétek a lehetséges eseteket, az *algoritmus* kidolgozása következik.

Az algoritmus arab eredetű szó. A IX. századbeli Ibn Musza al Hvárizmi, a híres arab matematikus nevéből származik. Azt az eljárást jelenti, amelynek a végrehajtásával megoldjuk a feladatot.

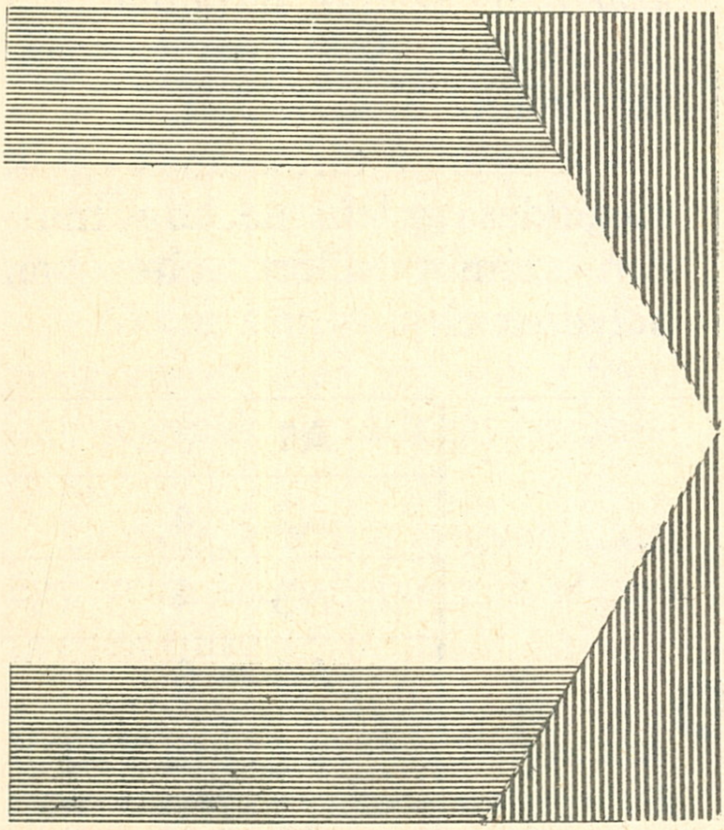
– Ez a folyamatábrához hasonló dolog lehet – tűnődött hangosan Csaba.

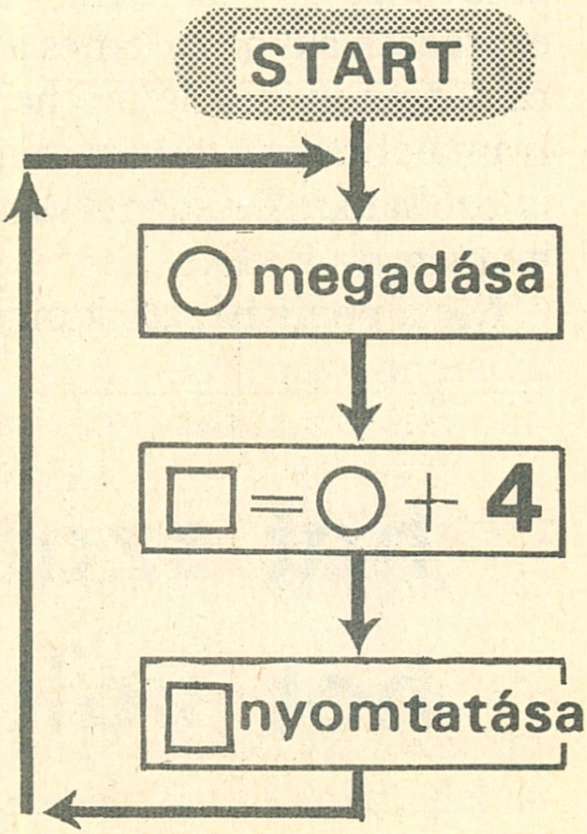
– A folyamatábra az algoritmus képe. Ha lerajzolod, könnyen tudod követni az algoritmus végrehajtását, és a hibákat is hamarabb felfedezed.

Itt van például az én hibás programom algoritmusa.

„Add meg a ○-t, adj hozzá négyet, és az eredményt nyomtasd ki. Utána kezdd elölről az egészet!”

Az algoritmus alapján készítettem a folyamatábrát:





– Persze hogy nem áll meg soha a program, hiszen ebben a folyamatábrában nincs is STOP! – szólt közbe Kinga.

– Nagyon jó szemed van! Ez az algoritmus bizony végtelen, s mivel hűen programoztam, a program sem ér véget soha, hanem végtelen ciklusban ismétlődik. Egy algoritmus legyen véges, pontos és minél rövidebb! Ezt tartsátok szem előtt.

– Mikor lehet elkezdeni programozni?

– A *programozás* a folyamatábra alapján történik. Éppen ezért csak akkor érdemes a programozásnak nekilátni, ha a folyamatábra már készen van, és ki is próbáltátok néhány számmal, hogyan működik.

Most pedig nézzünk meg egy-két gyakorlati programozási fogást, biztosan hasznát veszitek.

Írjunk programot az átlag kiszámítására. Legyenek, mondjuk, a magyarból kapott osztályzatok egy A vektorban, amelynek N eleme van.

Kiszámítandó az átlag.

– Máris készen vagy, Kinga?

– Dehogy! Nem tudom mi az a *vektor*.

– Igazad van, ezt még nem tanultátok. A vektor egy névvel ellátott számsorozat, amelyről tudom, hogy hány eleme van, azaz hány számot tartalmaz.

– Például, ezek a magyar osztályzataim: 3, 4, 4, 5, 4, 5.

– A számok tárolásának egyik megoldása az lehetne, hogy minden számot külön változóban helyezek el:

hat szám-hat változó

A1 = 3
A2 = 4
A3 = 4
A4 = 5
A5 = 4
A6 = 5

Ekkor persze az átlagot számító kifejezésben valamennyi változó nevet ki kell írnom:

$$AT = (A1 + A2 + A3 + A4 + A5 + A6)/6$$

(AT az átlag)

Ez pedig a programja:

```
10 LET A1 = 3
20 LET A2 = 4
30 LET A3 = 4
40 LET A4 = 5
50 LET A5 = 4
60 LET A6 = 5
70 LET AT = (A1+A2+A3+A4+A5+A6)/6
80 PRINT „ATLAG =”; AT
90 END
```

– És ha kapok egy újabb jegyet, akkor mit csinálok?

– Akkor sajnos javítani kell a programon, mert ez csak a fenti hat jegynek az átlagát számítja ki.

Ugyanezt a hat jegyet most egy vektorban helyezzük el, ezt így jelöljük:

A = [3, 4, 4, 5, 4, 5].

Ezek közül most valamelyik számot – azaz a vektor valamelyik elemét – úgy tudom kiválasztani, ha megadom a vektor nevét és utána zárójelben a kívánt szám sorszámát.

Például A(1) – olvasd á egy – az első számot, a 3-ast jelenti, A(4) – olvasd á négy – pedig az 5-öst, a negyediket. Általánosságban ezt úgy lehet írni: A(İ).

A(İ) az A vektornak annyiadik elemét jelenti, amennyi az İ változó értéke.

Például, ha İ = 1, akkor A(İ) = 3, mert a vektorunk első eleme a hármas.

Ugyanígy kell ezt írni a BASIC-ben is, mielőtt azonban egy kifejezésben használnád, deklarálni kell.

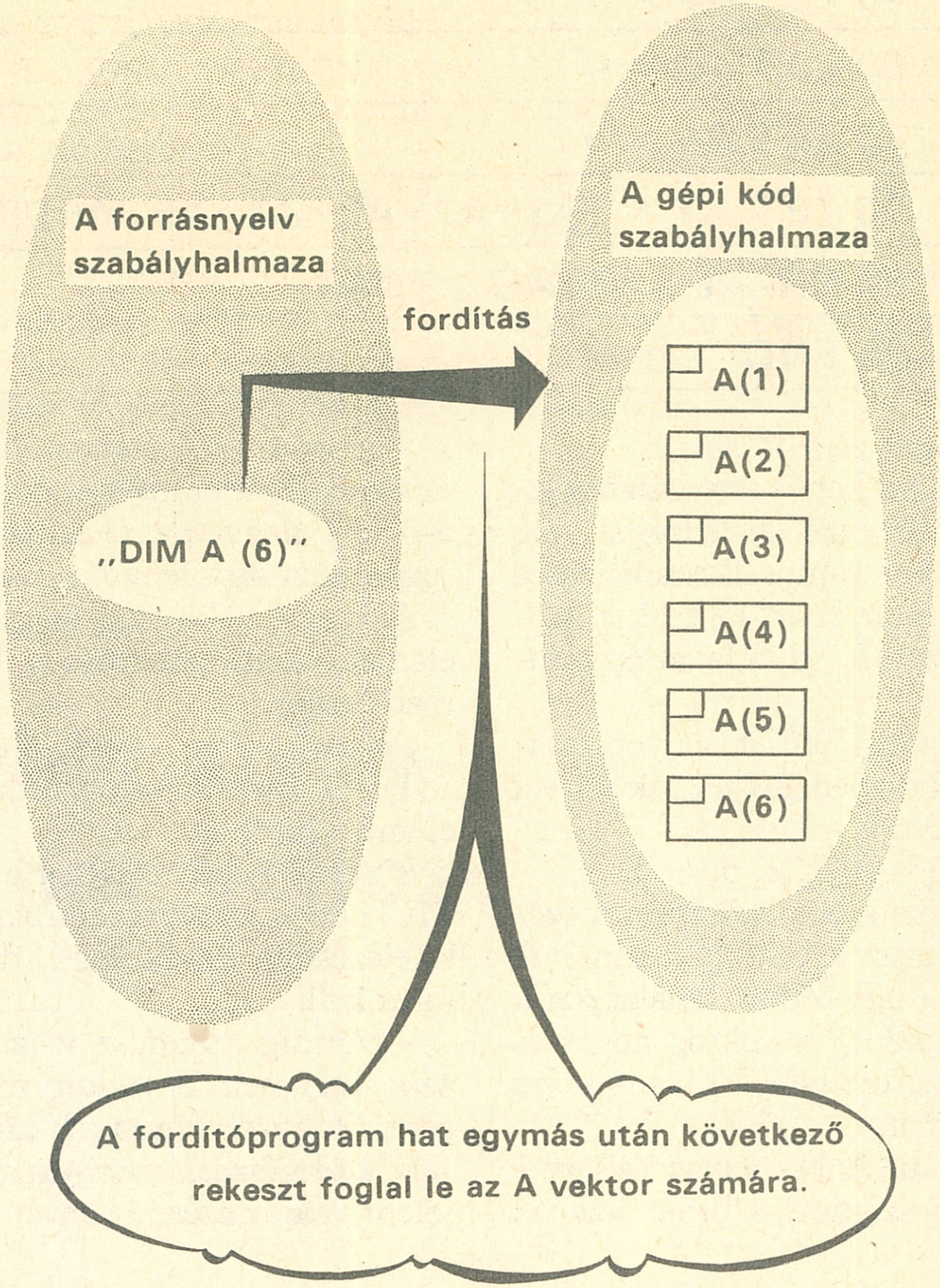
– Mindig mondasz valami új szót, már nem is tudom megjegyezni! – zsörtölődött Kinga.

– A *deklaráció* magyarul azt jelenti: kijelentés.

A fordítóprogram számára ki kell jelentened, hogy a vektornak mi lesz a neve, és hány elemet fog tartalmazni. Ez is BASIC utasítás, a kulcsszava a *DIM;*

Például a DIM A(6) azt jelenti, hogy az A nevű vektornak hat eleme lesz.

Amikor a fordítóprogram a DIM utasításhoz ér, annyi egymás után következő rekeszt foglal le a vektor számára, ahány eleme van.



Ebből következik, hogy a DIM utasításnak a programban mindig meg kell előznie a vektor használatát.

– Ez a vektor is csak hat osztályzatra lesz elég!

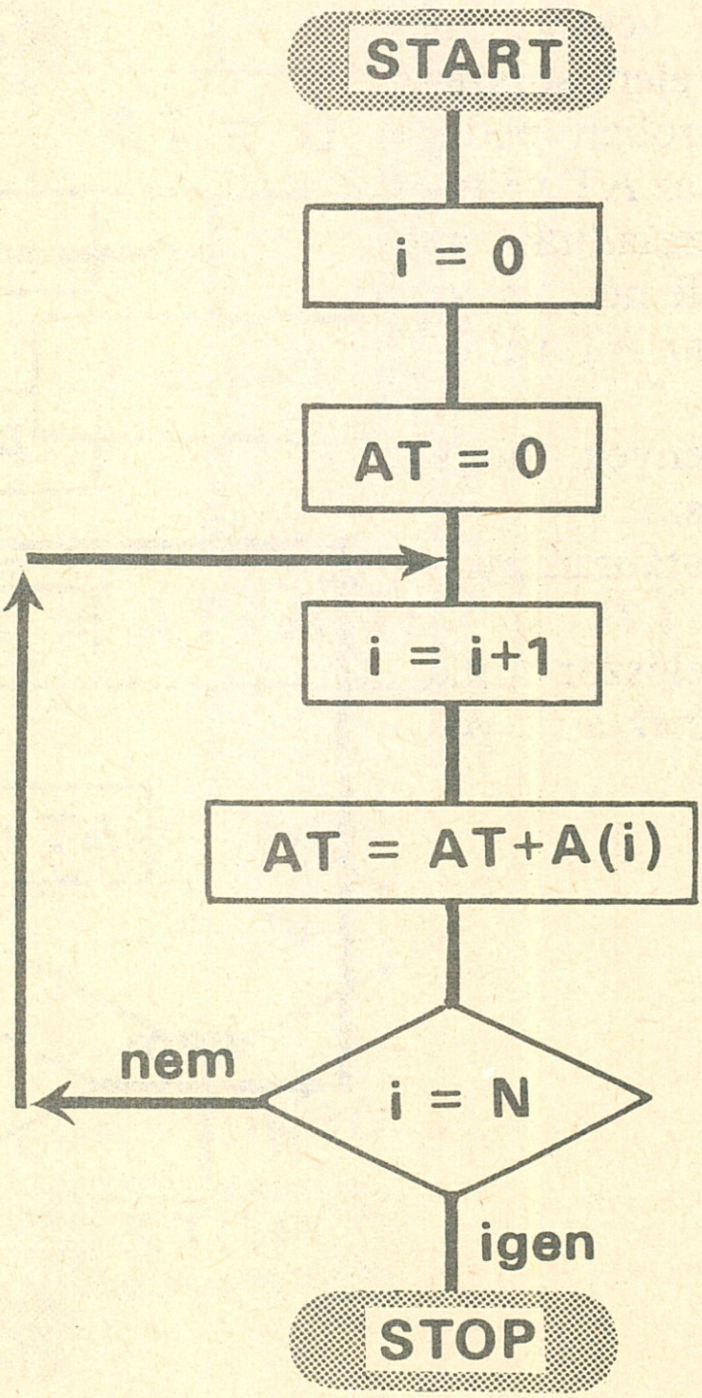
– Meg kell gondolni, Kinga, hogy hány osztályzatod lehet magyarból, és akkora, vagyis elég nagy vektort kell deklarálni.

– Ötven biztosan nem lesz.

– Rendben. Most pedig megmutatok egy új fogást, „felgöngyölítés" a neve.

Göngyölítünk? Göngyölítünk!

Gyerekek, most nézzétek meg alaposan ezt a folyamatábrát:



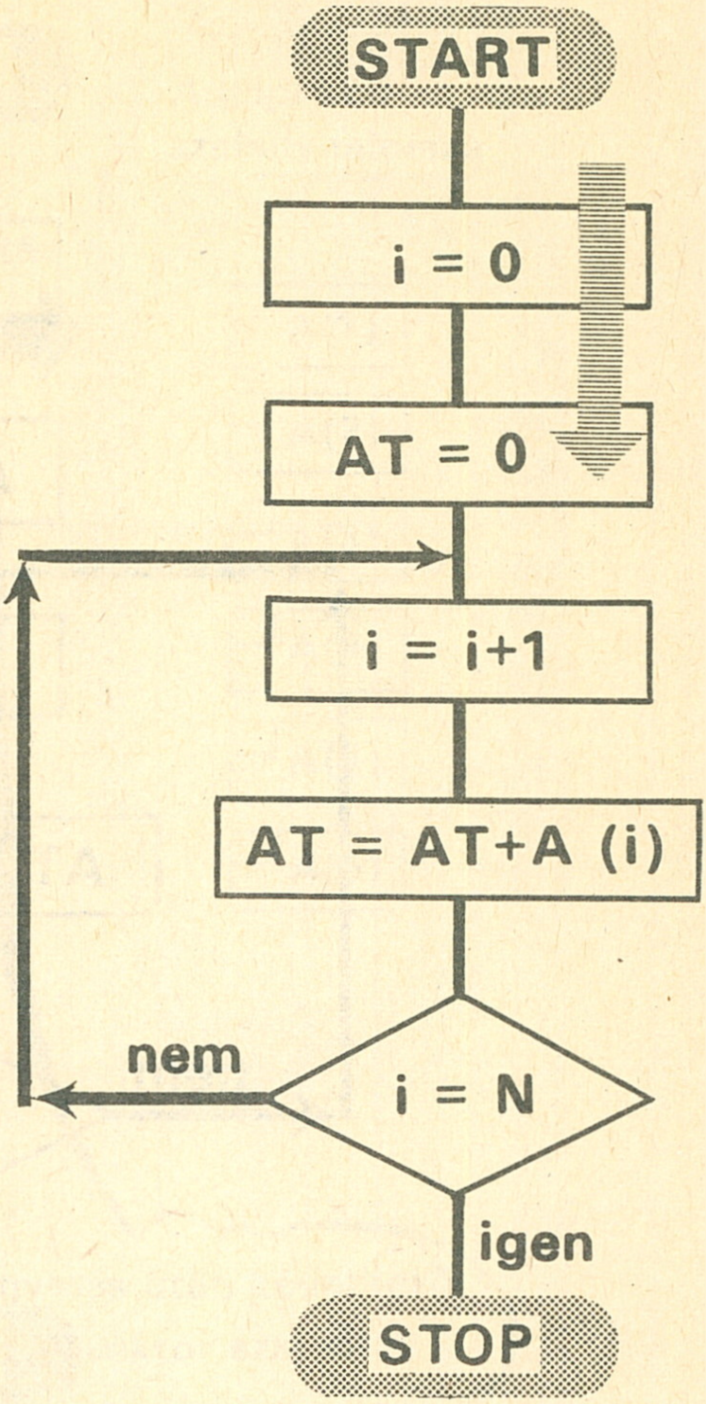


– Négy változónk van, İ, AT és az N elemű A vektor. Az İ változó jelenti az elem sorszámát, az AT változóban pedig „felgöngyölítünk": az AT változó értékéhez hozzáadjuk az A vektor İ-edik elemét, és az eredményt megint az AT változóba tesszük vissza.

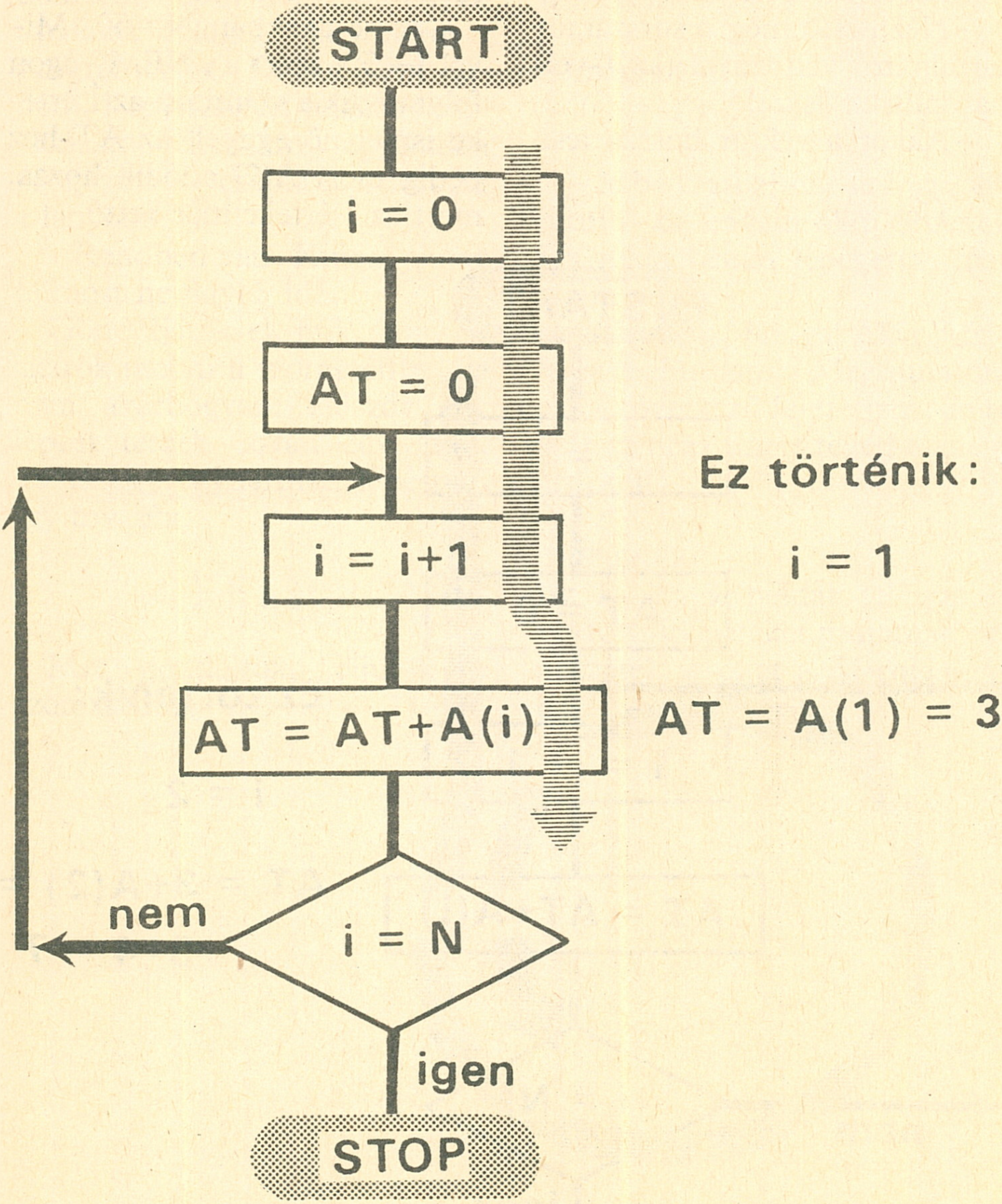
– Szóval az a lényeg, hogy ugyanoda kerül vissza.

– Igen. A folyamatábrán nyomon követheted.

Az İ és az AT először nulla értéket kap (figyeljétek a … vonalat!):



Ezután az İ változó értékét növeljük 1-gyel, az AT változóét pedig A(İ)-vel, azaz mivel İ = 1, A(1) értékével.



Most jön a vizsgálat: İ egyenlő-e N-nel?

– Mi az N? – kérdezte Balázs.

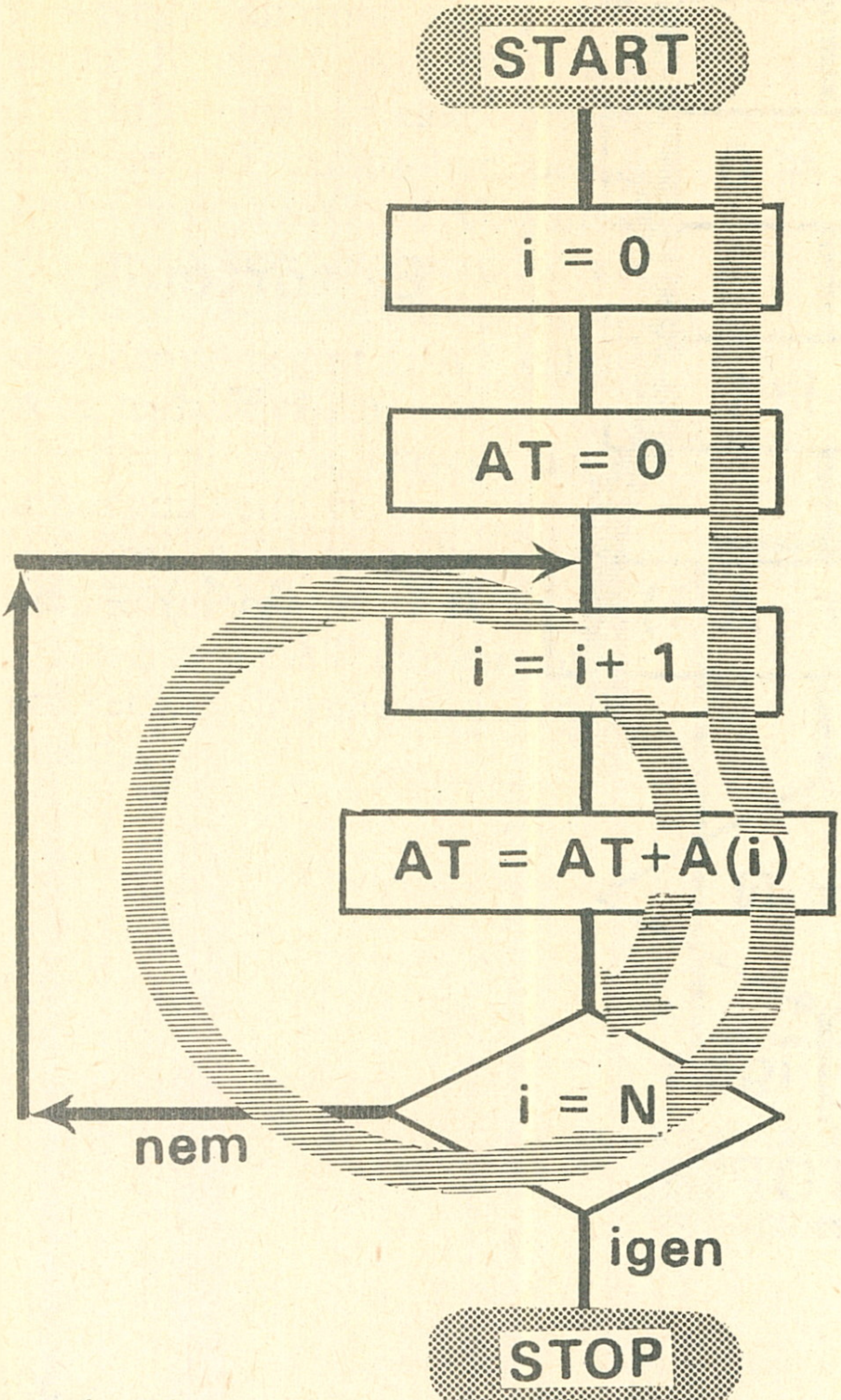
– Nézd csak meg: a sorszámot tartalmazó változóval (az İ-vel) hasonlítjuk össze!

– Akkor a vektor hossza lesz az!

– Úgy van.

– Az N csak azt jelentheti, hogy mekkora a vektor hossza, azaz hány elem van benne.

– A példánkban N = 50. Mivel İ ≠ 50, ezért a „NEM" ágon folytatódik a számítás: az İ értéke ismét nő eggyel, az AT-hez pedig most A(2) adódik hozzá, mert az İ értéke már kettő.



Ez történik:

i = 2

AT = 3+A(2) =
3+4 = 7

– És ez így fog menni, amíg az İ végre eléri az 50-et. Vedd tudomásul, apu, én nem ilyen unalmas dolognak képzeltem a programozást. És azt sem tudom, hogy milyen feladat megoldására lehet írni egyáltalán programot! – csapta le a ceruzát a lányom.

– Rendben van, Kinga, akkor te most ne figyelj ide.

Gondoljátok el, fiúk, hogy át akarok menni a szomszédba, de nem akárhogyan: először egy lépést teszek, aztán egy felet, utána egy egyharmad lépést és így tovább:

$$1, \tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{4}, \tfrac{1}{5} \ldots$$

Mit gondoltok, eljutok a szomszédba?

– Össze kell adni a számokat, és megtudjuk, mennyi utat lehet megtenni.

A megtett út

$$(s) = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \tfrac{1}{5} \ldots$$

– A bejárati ajtóig se fogsz eljutni – de engem nem érdekel. Ha ilyen bénán lépegetsz, magadra vess!

– Tévedsz, Kinga! Akár Afrikába is eljuthatok! Megmutatom ezt a következő fejezetben, de te bele se nézz, mert téged ez nem érdekel!

GYEREKEK!

Az 1., 2., 3. példák jellegzetes vektorműveleteket mutatnak be, nézzétek át őket a *Megoldások*-ban.

– Tudod, apa, azért nincs teljesen igazad. Azt mondtad, hogy a számítógép a barátunk. Hát milyen barát az, akinek mindent nekünk kell megcsinálni, hogy végre ő is tegyen valamit? – tört ki az elkeseredés Balázsból.

– Csak részben van igazad. Nem mindent, csak a programot kell megcsinálnod. Ha kész a program, a számítógép annyiszor végrehajtja, ahányszor csak akarod, méghozzá hibátlanul. Soha nem fárad el, soha nem panaszkodik. Vajon melyikünk tudná ebben utánozni???

Nézzétek csak meg ezt a lépegetős feladatot!

Ha nekem kellene kiszámolni...

Hová jut el az az ember, aki így lép: először egyet, aztán egy felet, aztán egy egyharmad lépést és így tovább:

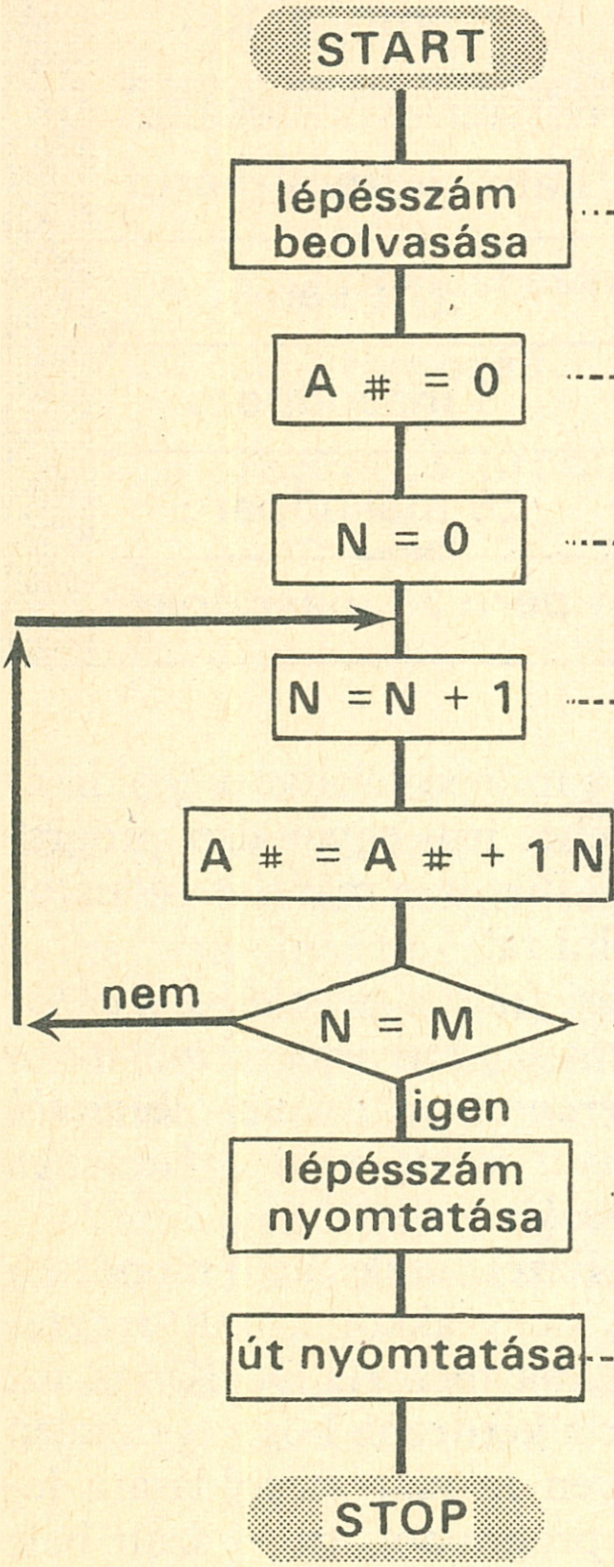
$$1, \tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{4}$$

Ne törjétek rajta a fejeteket, akárhová el lehet így menni annak ellenére, hogy a lépések egyre kisebbek lesznek.

– Egyszerűen ki lehet számítani a megtett utat – mondta Csaba –, de nagyon sok időbe telik.

– Én 10 ezer lépésig kiszámol-

tam egy HT 10807 iskola-számítógéppel. Itt a folyamatábra és a programja:




```
10 INPUT M
20 LET A # = 0
30 LET N = 0
40 LET N = N+1
50 LET A # = A #+1/N
60 IF N = M THEN
        GOTO 80
70 GOTO 40
80 PRINT M
90 PRINT A#
100 END
```

Ezek voltak az eredmények:

lépésszám	úthossz (m)	számítási idő
1	1	nem tudtam mérni
10	2,92896	(itt sem)
100	5,18737	3 másodperc
1000	7,48547	26 másodperc
10 000	9,78760	4 perc 20 másodperc

– Igazad van, apa – mondta megtörten a lányom, mert mégiscsak idehozta a kíváncsiság. – Én ezt két nap alatt sem tudtam volna kiszámítani.

– Kinga, ha 1 perc alatt számítasz ki egy úthosszt, és éjjel-nappal dolgozol, akkor is több mint egy hetedbe kerülne ez a számítás.

– És még hibázhat is közben! – kiáltotta Balázs.

– Könnyen megeshet az is. Örülök, hogy észrevettétek a számítógép fő erényeit: gyorsan dolgozik, fáradhatatlan, pontos, és éppen azt csinálja, amit kell. Ugye, azt kérdezted, Kinga, hogy milyen feladat megoldására lehet írni egyáltalán programot? Biztosan már ti is tudnátok felelni rá:

Egy feladat megoldására akkor érdemes programot írni, ha a program elkészítése kevesebb munkát jelent, mint a közvetlen számolás.

– Ezzel csak azt mondtad, apa, hogy akkor írjunk programot, ha megéri. De Kinga azt akarta kérdezni, hogy egyáltalán milyen feladat megoldására lehet programot írni – szólt bele a nagyfiam, aki időnként védelmébe szokta venni a saját húgát is.

– Ez a dolognak a másik oldala. Emlékeztek a feladatmegoldás sorrendjére?



Az algoritmuskészítés megelőzi a programozást, vagyis csak akkor lehet programozni, ha már van algoritmus.

Az a válaszom tehát, hogy minden olyan feladat megoldható számítógéppel, amelynek van algoritmusa.

– Van olyan feladat egyáltalán, amivel a számítógép nem boldogul?

– Hogyne volna, és látszólag milyen egyszerűnek tűnik!

Ami még egy számítógépnek is sok

– Biztosan versenyeztél már a barátoddal azon, hogy ki tud hamarabb felismerni egy közeledő autót.

– Dedós koromban játszottam ilyesmit utoljára – húzta fel az orrát Kinga –, de akkor is nagyon hamar felismertem.

– Szerencsére már nem vagy dedós, így meg tudod mondani, hogy miről is ismerted fel őket.

– Az alakjukról vagy a színükről.

– Kingácskám, az sokszor kevés. Van úgy, hogy csak a lámpák mások!

– Mit tudom én már, de felismertem őket, az biztos!

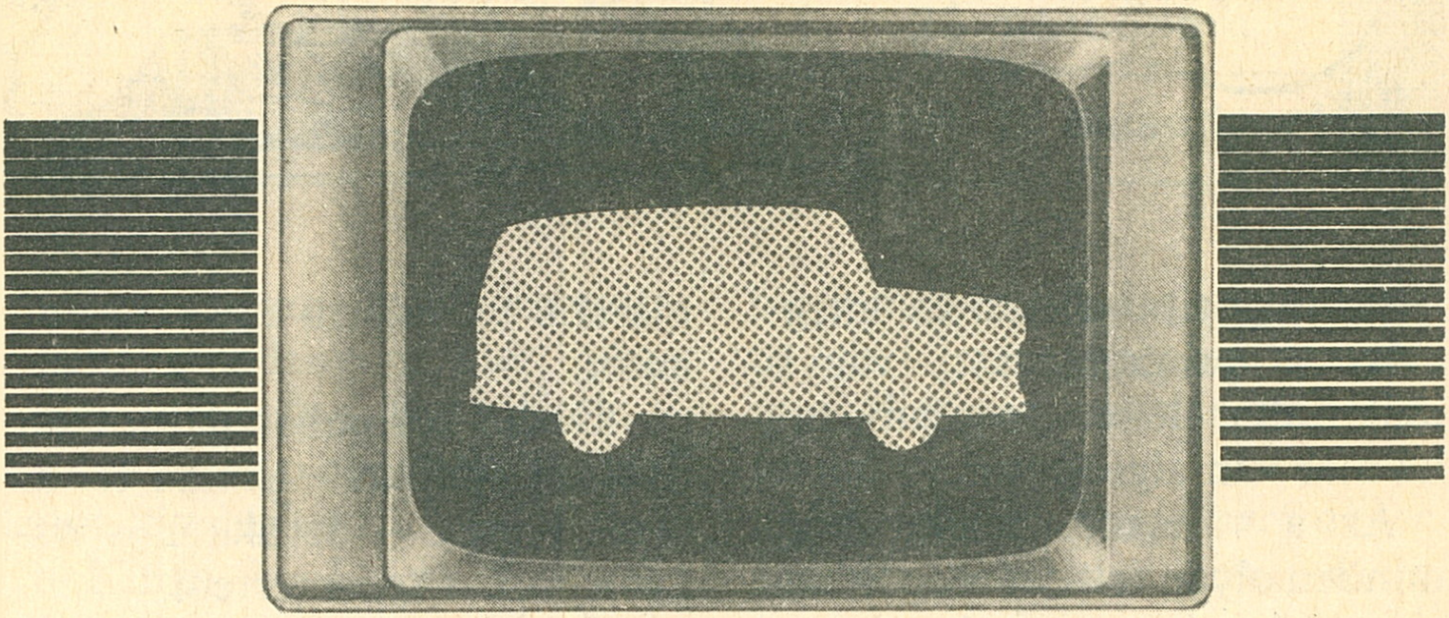
– Látjátok, ez a feladat nektek gyerekjáték volt, de a számítógép számára majdnem megoldhatatlan probléma.

– Nincsen algoritmusa?

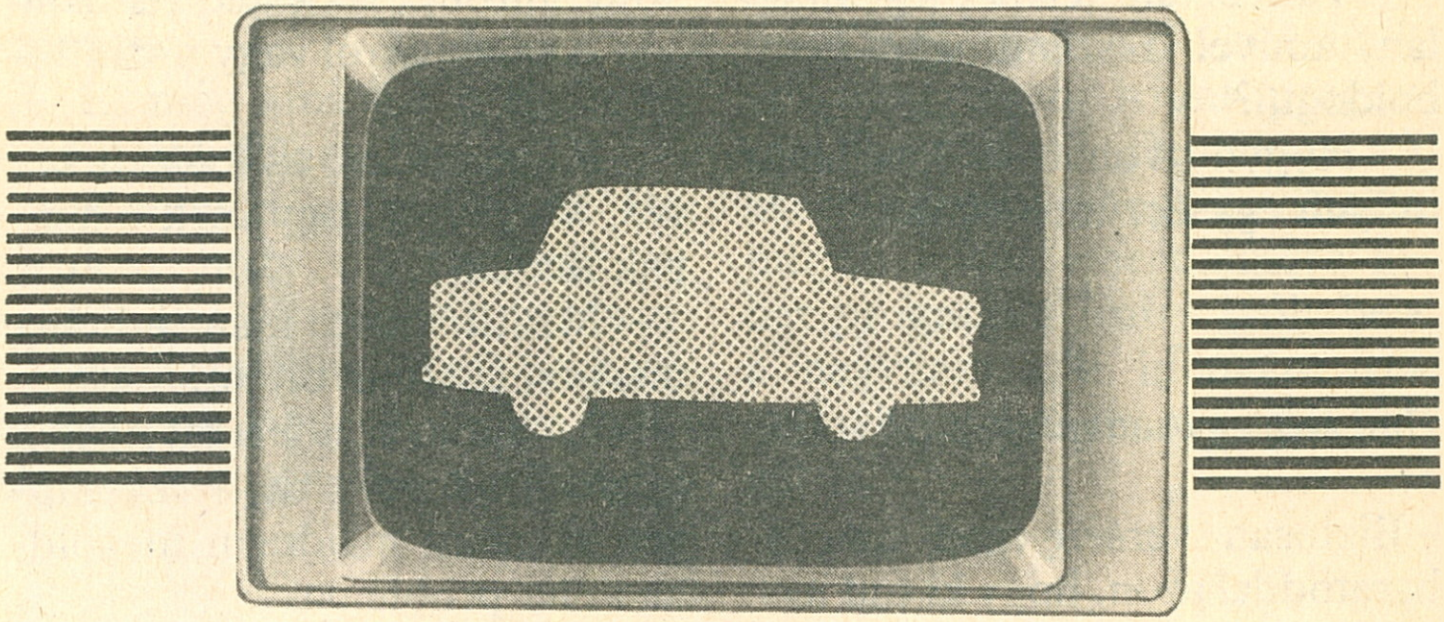
– Olyan még nincs, amelyiknek a programja gyorsaságban felvenné a versenyt veletek. Gondoljatok csak bele, hogy mennyi adatot észlel a szemetek a közeledő autóról: a mozgását, az alakját, a színét, az utasainak számát, az útfekvését, és még hosszan sorolhatnám a többit. Ezek közül azonban az agyatok gyorsan kiválasztja a lényeges adatokat.

– Úgy gondolom, a leglényegesebb adat az autó körvonala lehet. Sok esetben a körvonal jellegzetes, mint például egy Trabant Combi vagy egy Trabant Limousine esetében.

Trabant Combi



Trabant Limousine

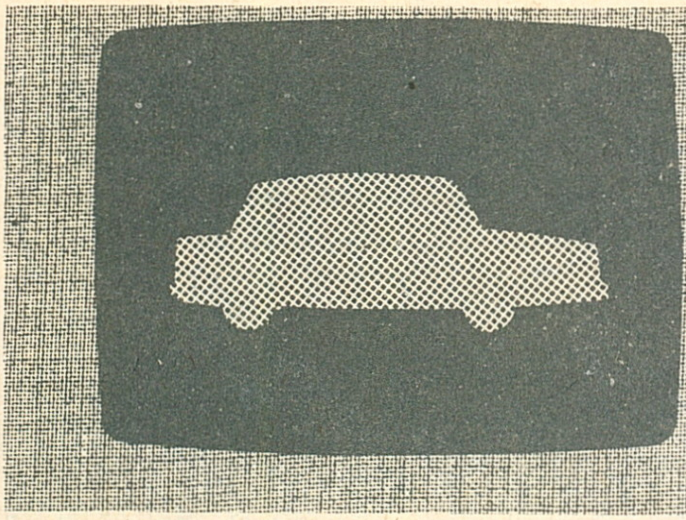


– Az lenne a jó, ha az autótípusok körvonalát tárolni lehetne a memóriában, és akkor csak a közeledő autó kontúrját kellene összehasonlítani a tárolt körvonalakkal.

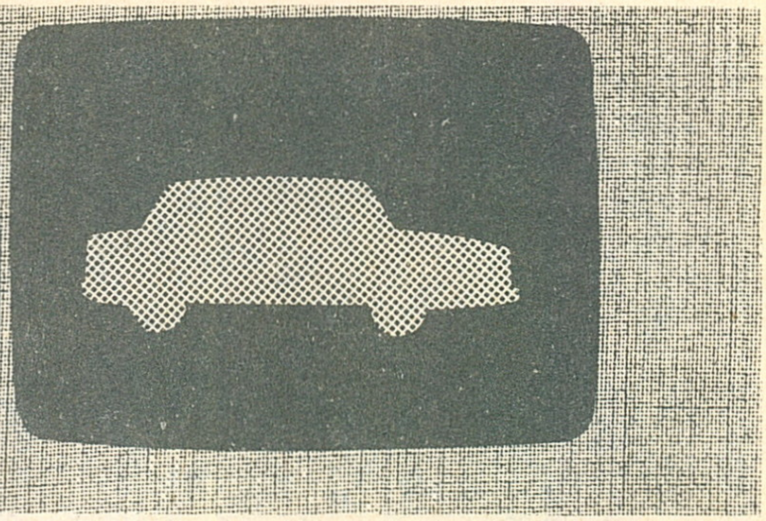
– Ezt az egyszerű gondolatot megfelelő algoritmussal nagyon nehéz megvalósítani.

Elég-e, ha csak az oldalnézeti körvonalakat hasonlítjuk össze?

Nem mindig!



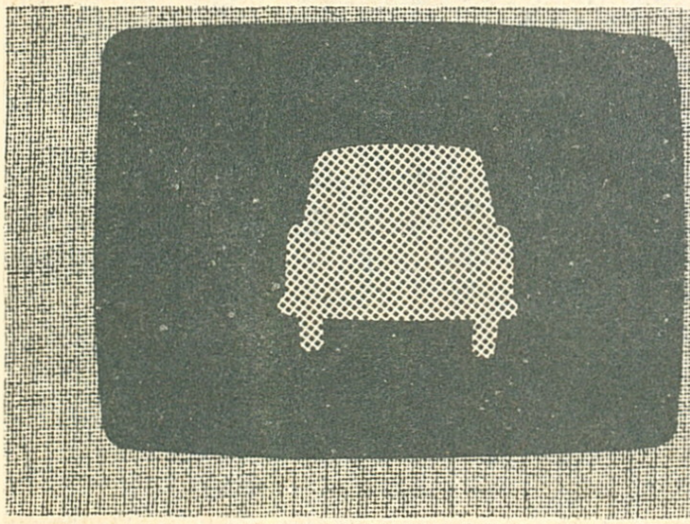
Skoda 120



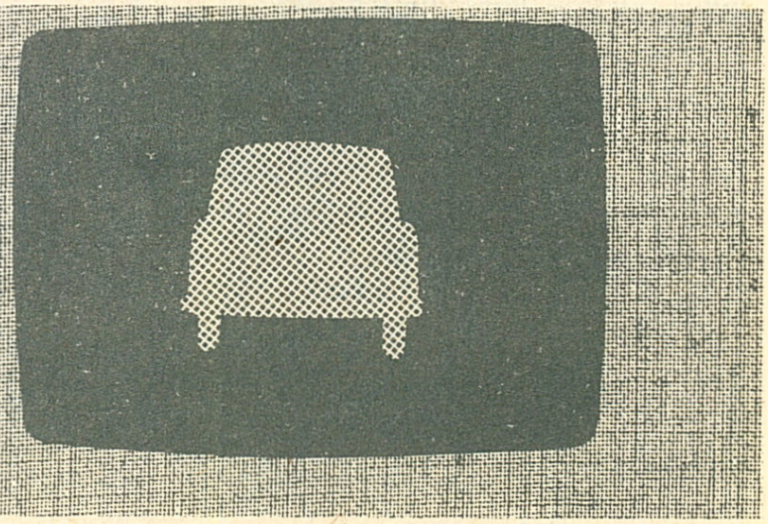
Skoda 105. A körvonalak megegyeznek, a típusok mégis különbözők

Vagy az elölnézeti körvonalakat használjuk?

Ez is lehet megtévesztő:



Trabant Combi elölről



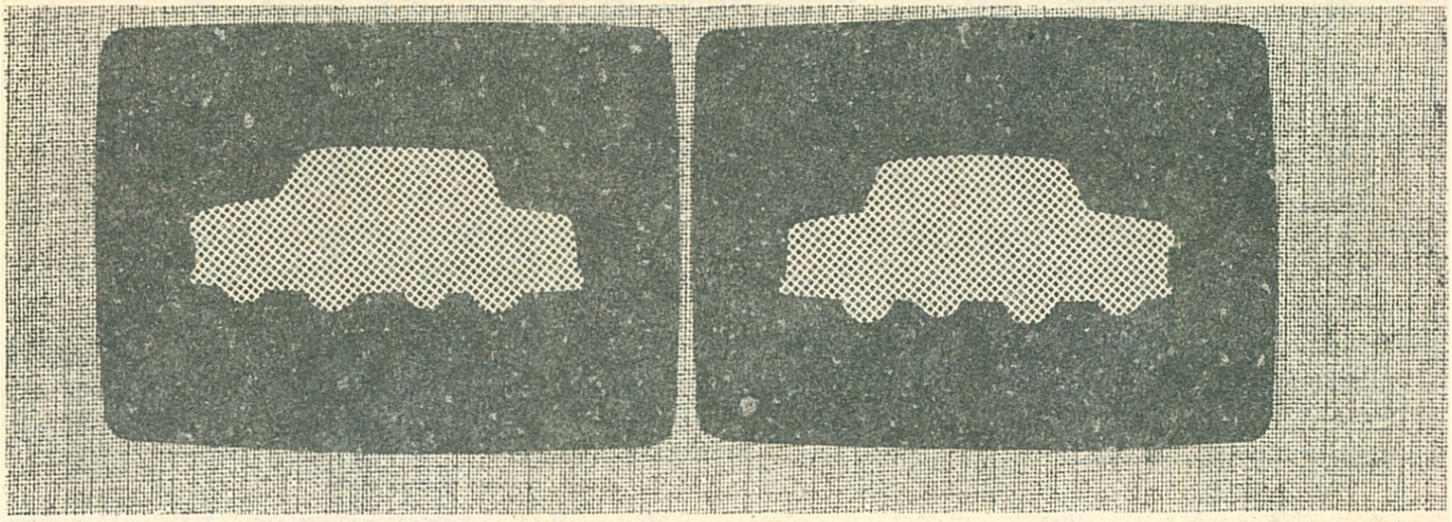
Trabant Limousine elölről. A körvonalak itt is megegyeznek, de a típusok eltérők

Az oldalnézeti és elölnézeti körvonalak használata külön-külön, látjátok, nem biztos.

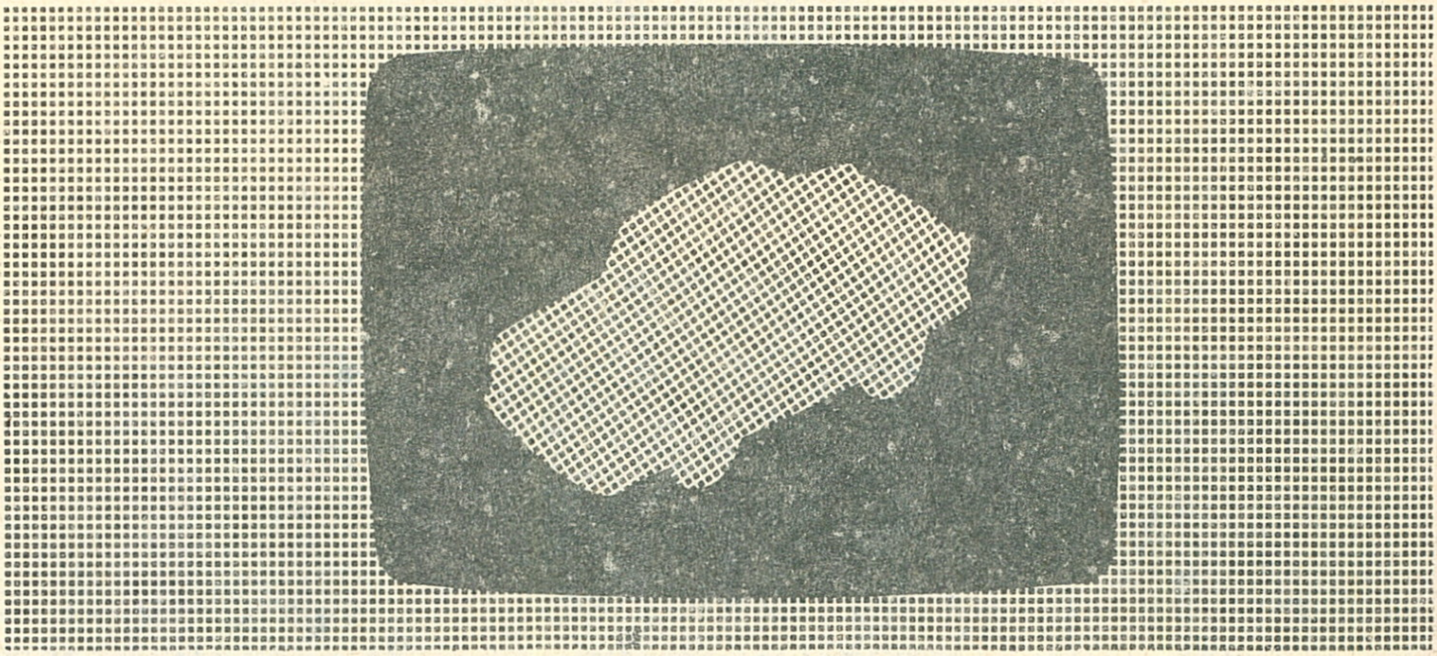
– Hát akkor használjuk őket együtt!

– Az a baj, Balázs, hogy a valóságban ritkán látod egy autónak pontosan az oldalnézetét vagy pontosan az elölnézetét. A gyakori az általános nézet.

Ahogy közeledik, fordul, vagy sávot vált, láthatod az elejét és a jobb oldalát vagy a bal oldalát és az elejét,

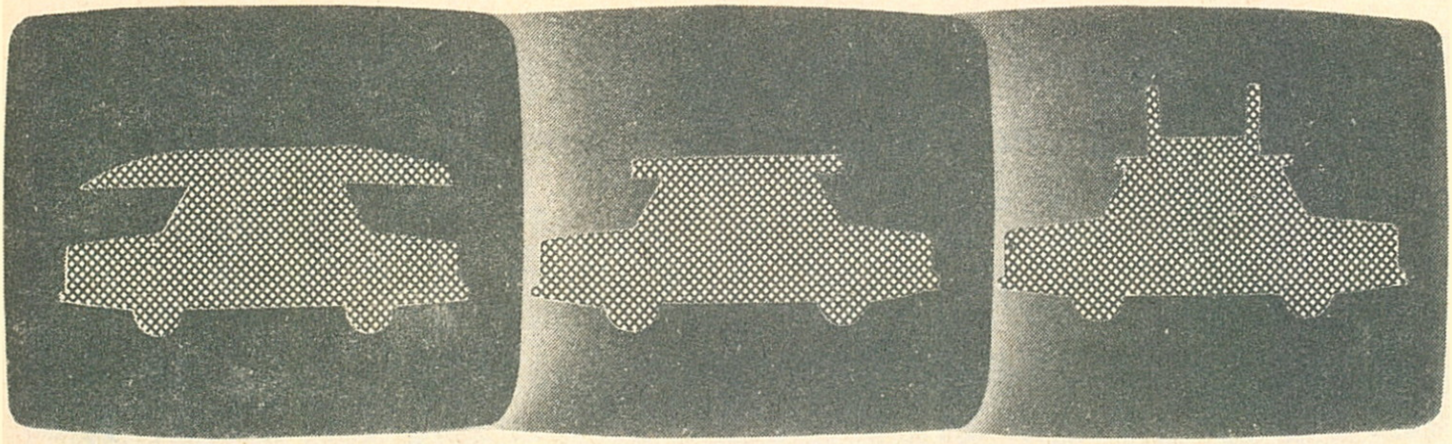


sőt egy út menti dombról nézve akár még a tetejét is:



– Akkor végtelen sok körvonala lehet egy autónak!

– Igen, ez az egyik gondunk. És ahogy egyre jobban, mélyebben vizsgáljuk a feladatot, újabb nehézségekkel találkozunk. Egy autón lehet tetőcsomagtartó és csomag is. Ezektől nem függ az autó típusa, de a kocsi körvonalát alaposan megváltoztatják.



Ez itt ugyanaz az autó egy csónakkal, egy pingpongasztallal vagy egy étkezőasztallal a tetején.

Száz szónak is egy a vége: láthatjátok, még csak a feladat megértésénél tartunk, és egyre bonyolultabb a helyzet.

De én nem is erről akartam végül beszélni nektek, hanem arról, hogy miért olyan fontos számunkra a számítógép.

A titok nyitja

Most már ti is tudjátok, hogy a számítógép adatok feldolgozását végző, programozható elektronikus gép.

Nem valami rejtelmes, gondolkodó gépagy, hanem olyan ember készítette eszköz, mint mondjuk egy kerékpár.

A kerékpárral megsokszorozod lábad gyorsaságát – a számítógép sok ezerszer gyorsabban számol adatok tömegével.

Igaz, azt a programot hajtja végre, amit te írtál meg – de hát a kerékpárt is hajtanod kell!

Van azonban egy lényeges tulajdonsága, és ez a sokcélúság vagy idegen szóval: az univerzalitás.

Egy kerékpár csak a helyváltoztatásod eszköze, másra nem használható.

A számítógép azonban BÁRMILYEN FELADAT PROGRAMJÁT VÉGREHAJTJA.

Ezért találod ott a repülőgép fedélzetén, ahol figyelemmel kíséri a repülés adatainak változását, és csak a legfontosabbakat közli a pilótával.

Vagy megtalálod a korszerű pályaudvarok vezérlőtermében, mert ott sok száz váltót, sorompót több ezer tehervagon mozgását kell felügyelni biztonságosan.

Ott vannak a számítógépek a közlekedésben, a mezőgazdaságban, az iparban, az oktatásban és a hétköznapjainkban is.

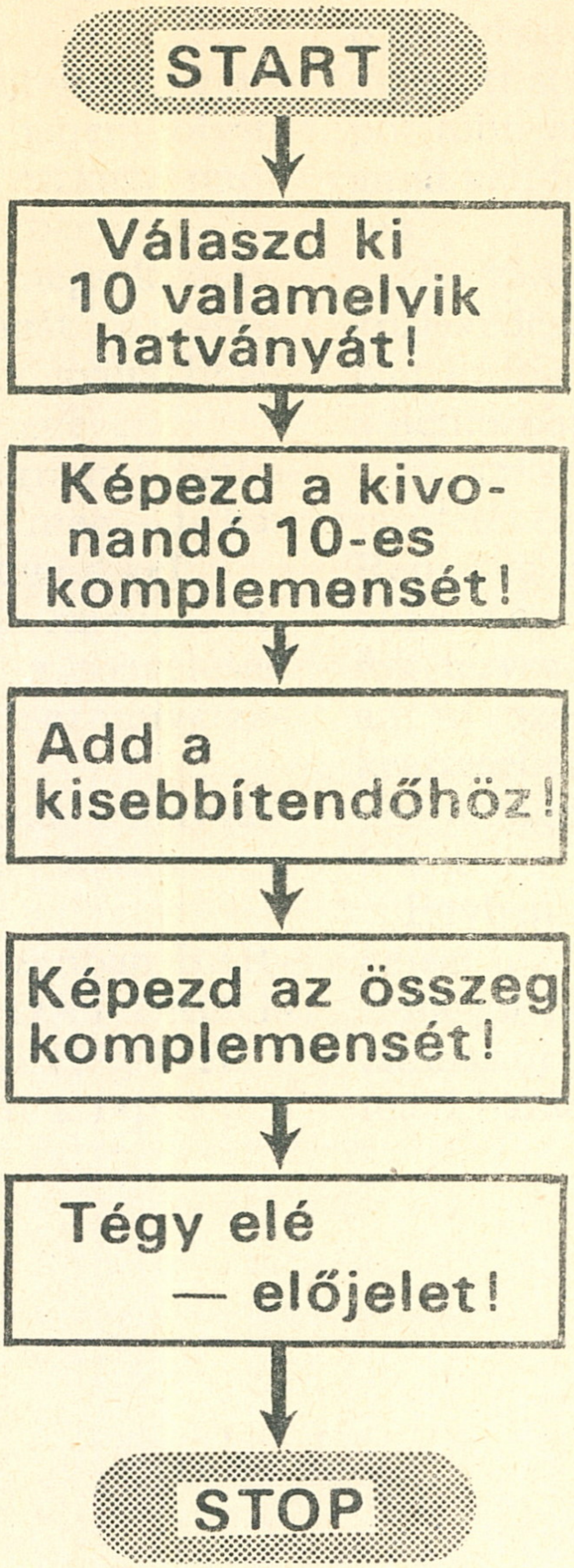
Persze számtalan felhasználási területet tudnátok ti is mondani. Rengeteg kérdés vár ma is megoldásra az űrhajózásban, a robotok tervezésében éppúgy, mint az új számítógépek megszerkesztésében. Ki tudja, talán éppen te fogod megoldani valamelyiket!

Egyben azonban biztos lehetsz:

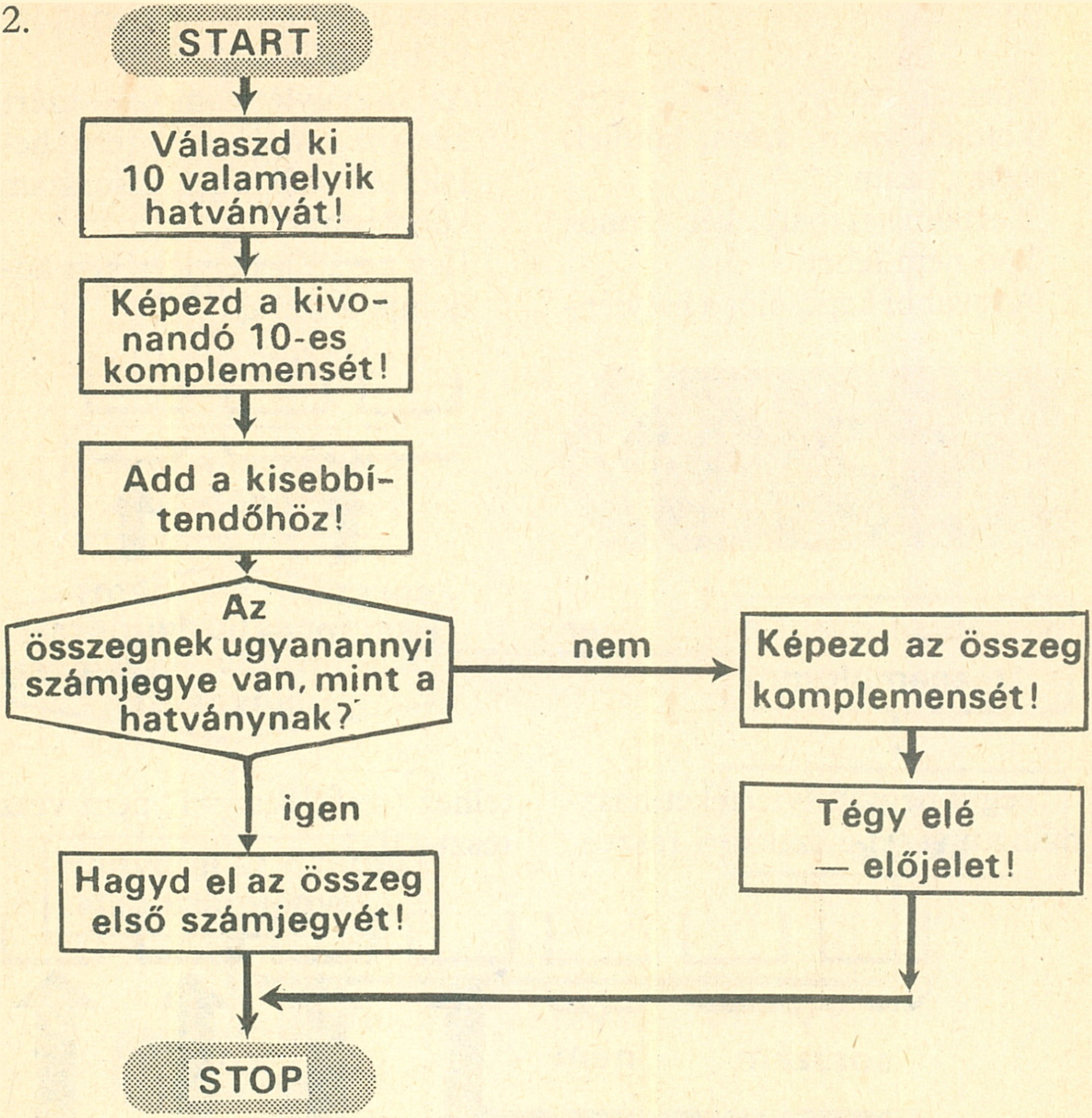
ha jól használod, és szívesen tanulsz új dolgokat, mindig segít majd barátod, *a számítógép!*

I. FEJEZET

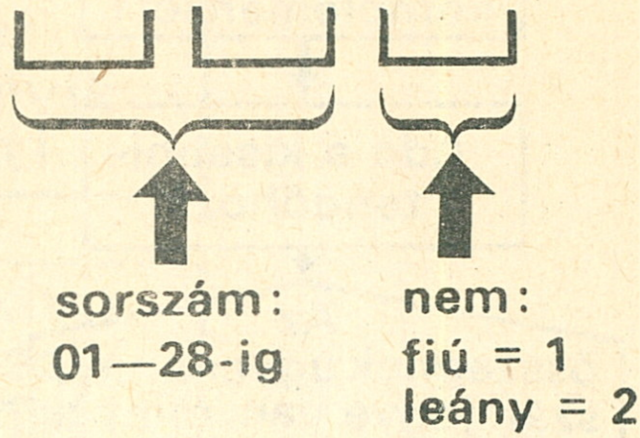
1.


START
Válaszd ki
10 valamelyik
hatványát!
Képezd a kivo-
nandó 10-es
komplemensét!
Add a
kisebbítendőhöz!
Képezd az összeg
komplemensét!
Tégy elé
— előjelet!
STOP

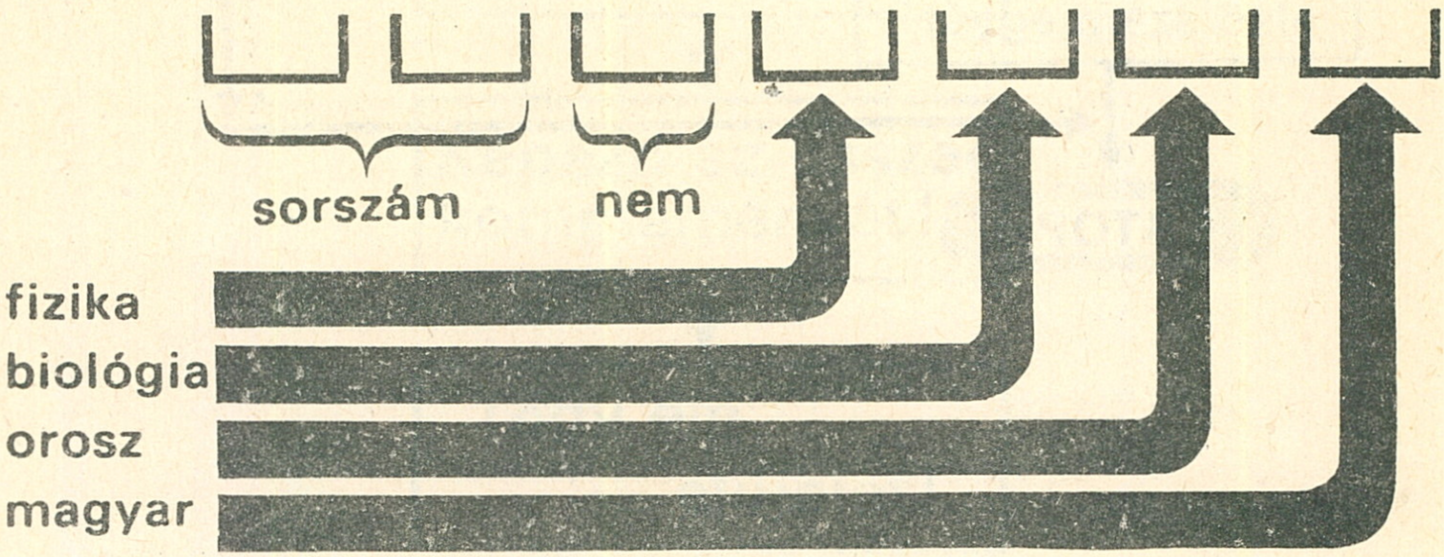
2.


START
Válaszd ki
10 valamelyik
hatványát!
Képezd a kivo-
nandó 10-es
komplemensét!
Add a kisebbí-
tendőhöz!
Az
összegnek ugyanannyi
számjegye van, mint a
hatványnak?
nem
Képezd az összeg
komplemensét!
Tégy elé
— előjelet!
igen
Hagyd el az összeg
első számjegyét!
STOP

1. Formai szabály: betű, betű, kötőjel, szám, szám, kötőjel, szám, szám
Tartalmi szabály: két azonos kód nem lehet.
2. A további kapcsolót a helyiérték-kapcsolók és az elem közé kell kötni.
3. Az osztálylétszám 26, ezért két tízes számrendszerű helyiérték szükséges a sorszám kódolására.
Egy további helyiértéken kódoljuk a nemet:



Végül négy helyiértéket használhatunk fel a szakköri részvételhez (részt vesz = 1, nem vesz részt = 0):



Így pl. az 1211101 (egy-kettő-egy-egy-egy-nulla-egy) kód az osztálynévsorbeli 12-ik tanulót jelenti, aki fiú, és tagja a fizika-, biológia- és magyarszakkörnek.

III. FEJEZET

1. Rövidebb kódot kapunk, ha a tizenhatos számrendszert használjuk.

   Változatlanul két helyiérték kell a sorszámnak, egy a nemnek, de csak egy az összes szakköri munkának!

   Négy szakkör van (négy helyiérték), bármelyikre jár vagy nem jár az illető (egy helyiértéken 0 vagy 1 állhat), az összesen 16 eset.

   Például a 1211101 kódból így lesz OC1D.
2. Megoldás:

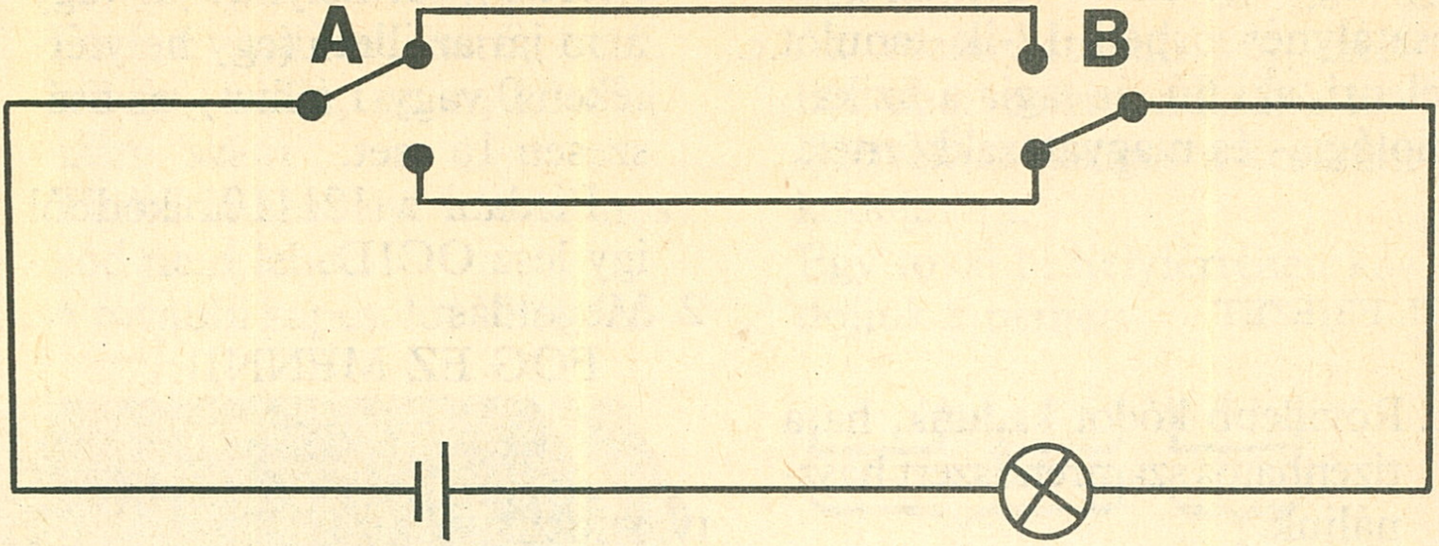
   FOG EZ MENNI!

IV. FEJEZET

1. A kétbemenetű KIZÁRÓ VAGY kapu igazságtáblázata:

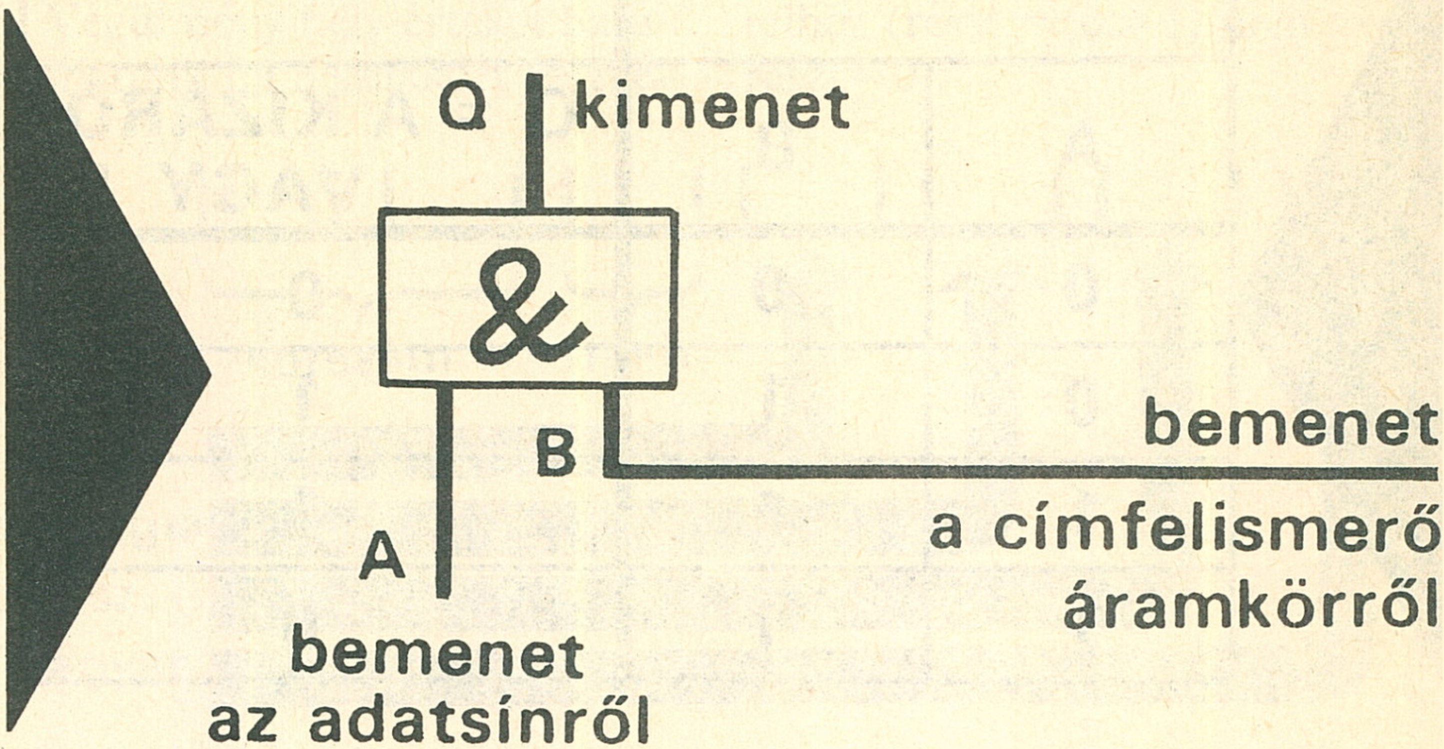
Bemenetek		Kimenet
A	B	Q = A KIZÁRÓ VAGY B
0	0	0
0	1	1
1	0	1
1	1	0

2. Az áramkör:

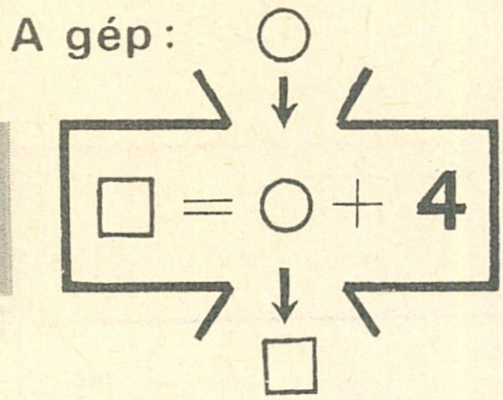


3. Az adatsorompó működésének megértéséhez elég egyetlen helyiérték vizsgálata:

Ha címegyezés van, a B bemenet igazzá válik, vagyis 1 lesz: az ÉS kapu csak ebben az esetben engedi tovább a Q kimenetre az A bemenet 1 jelét.



1. A folyamatábra jobb oldalán lévő vékony nyilak megmutatják, hogy melyik folyamatábra-elemnek melyik BASIC utasítás felel meg. A □-nek a „NE”, a ○-nek „KO” változót használtam. A nullát áthúzással jelöljétek!

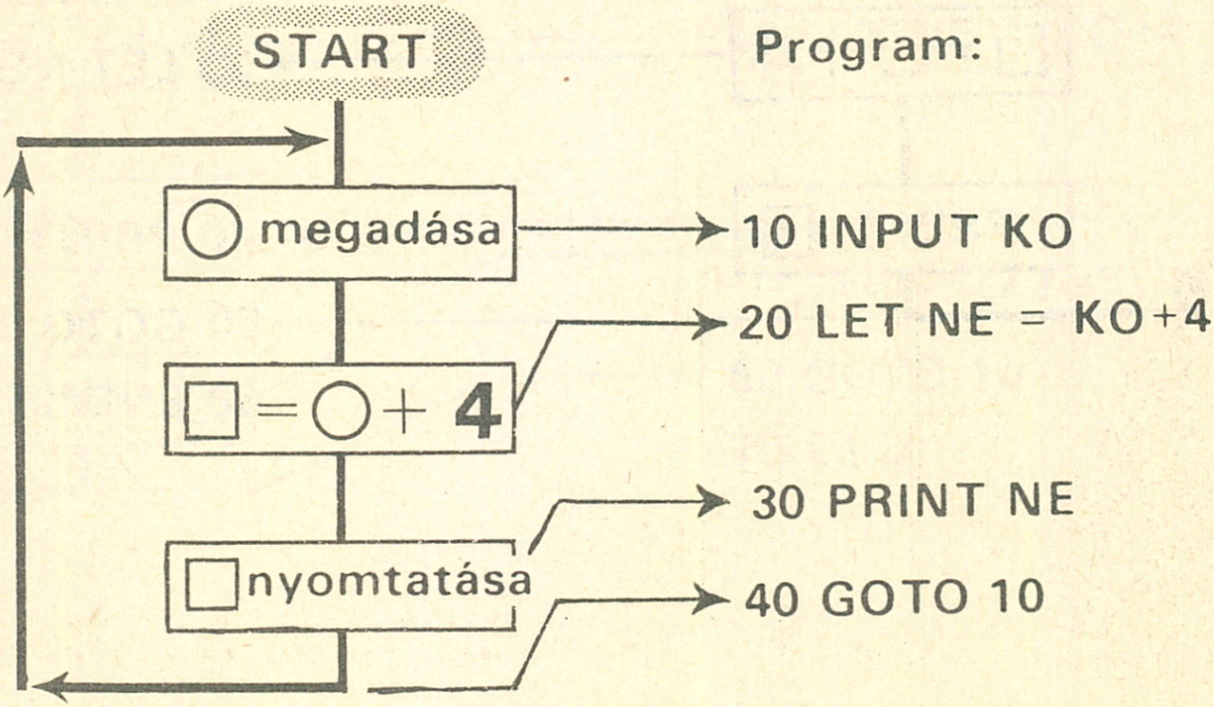


[redacted]

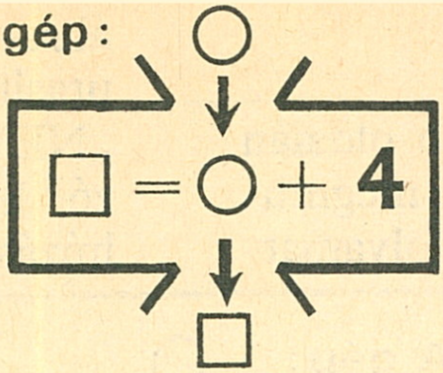
[redacted]

Folyamatábra:

[redacted]

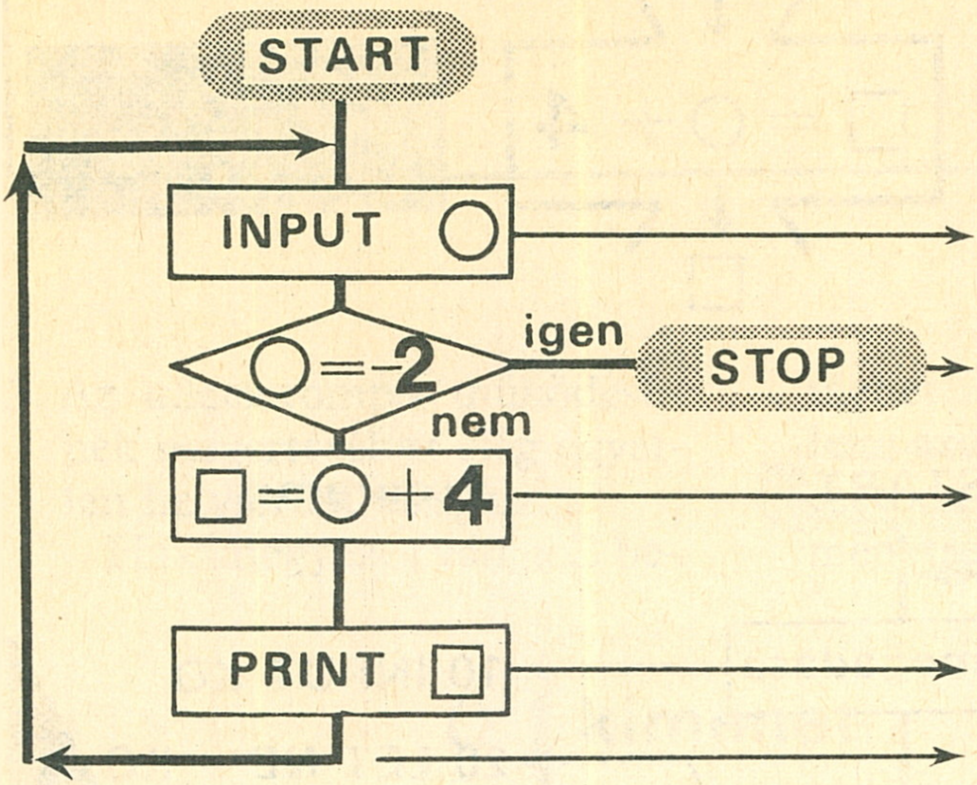


2. A gép:



ha ○ = −2
a gép megáll.

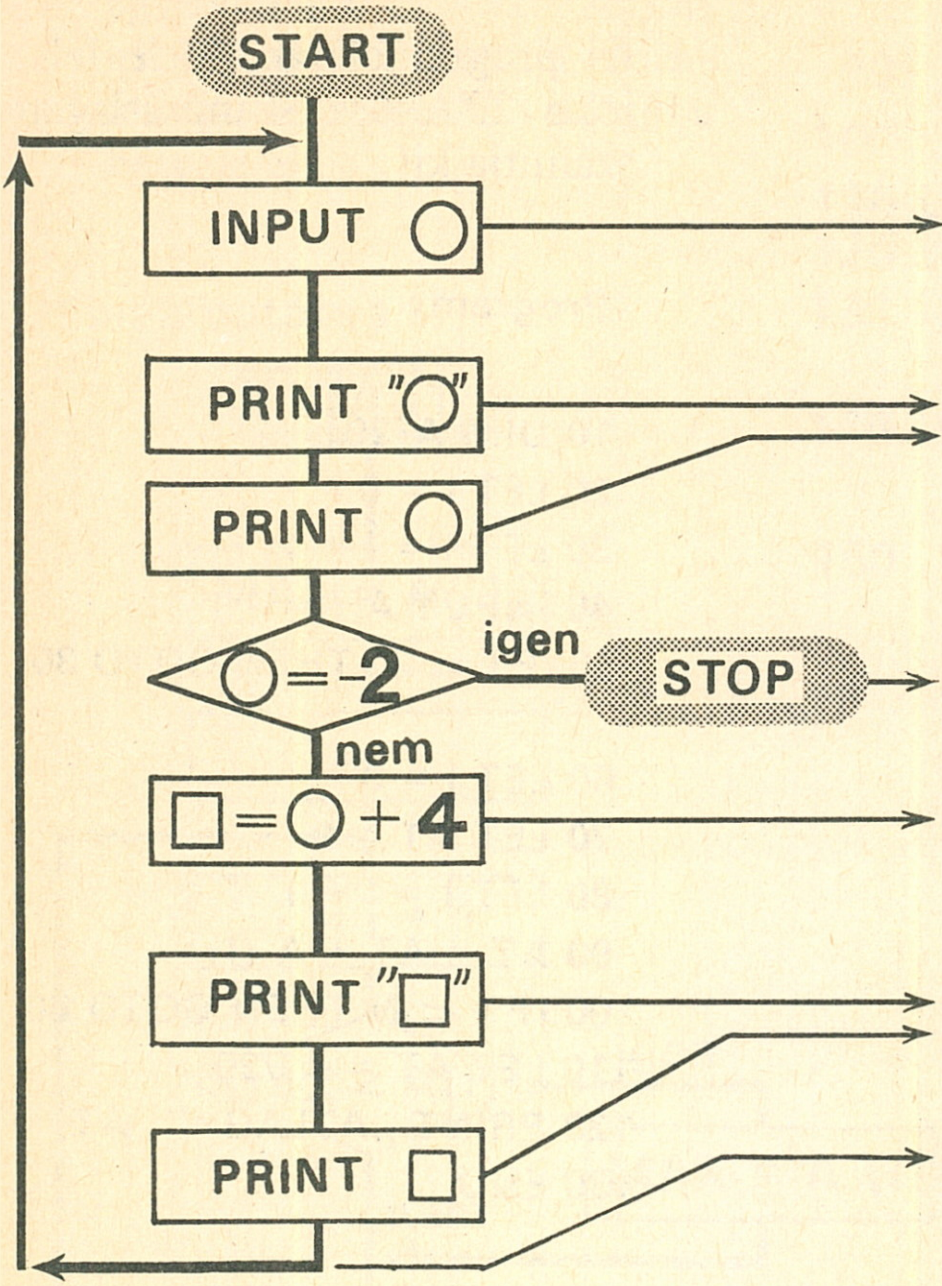
Folyamatábra:



Program:

```
10 INPUT KO
20 IF KO = —2
   THEN GOTO 60
30 LET NE = KO+4
40 PRINT NE
50 GOTO 10
60 END
```

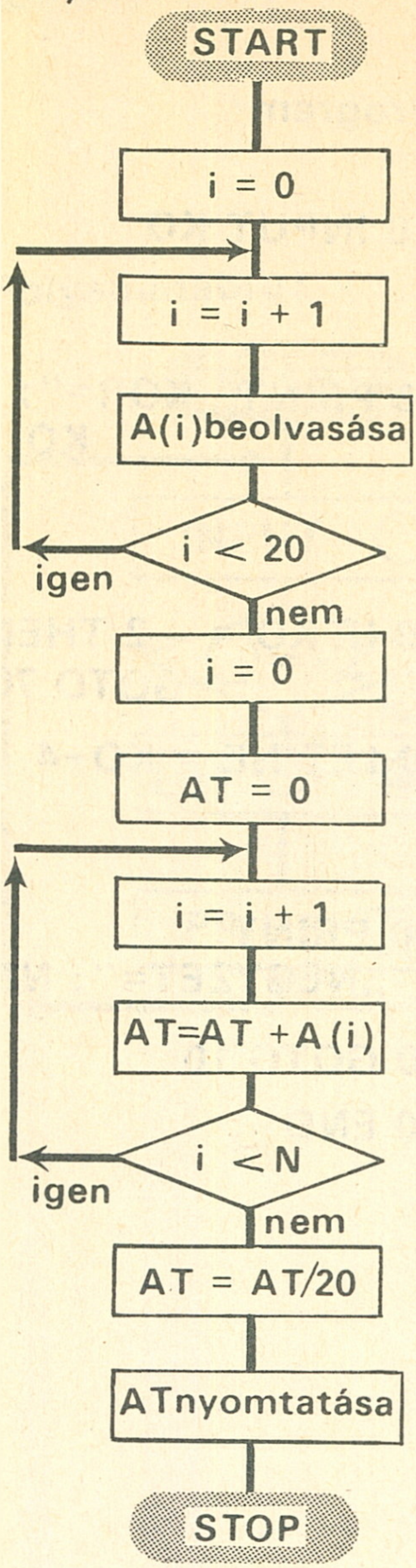
Folyamatábra: 3.



Program:

```
10 INPUT KO
20 PRINT „KOR=";
   KO
30 IF KO = —2 THEN
   GOTO 70
40 LET NE = KO+4
50 PRINT
   „NEGYZET="; NE
60 GOTO 10
70 END
```

Folyamatábra:



1. Írj programot, amely legfeljebb 20 egész szám átlagát számítja ki!

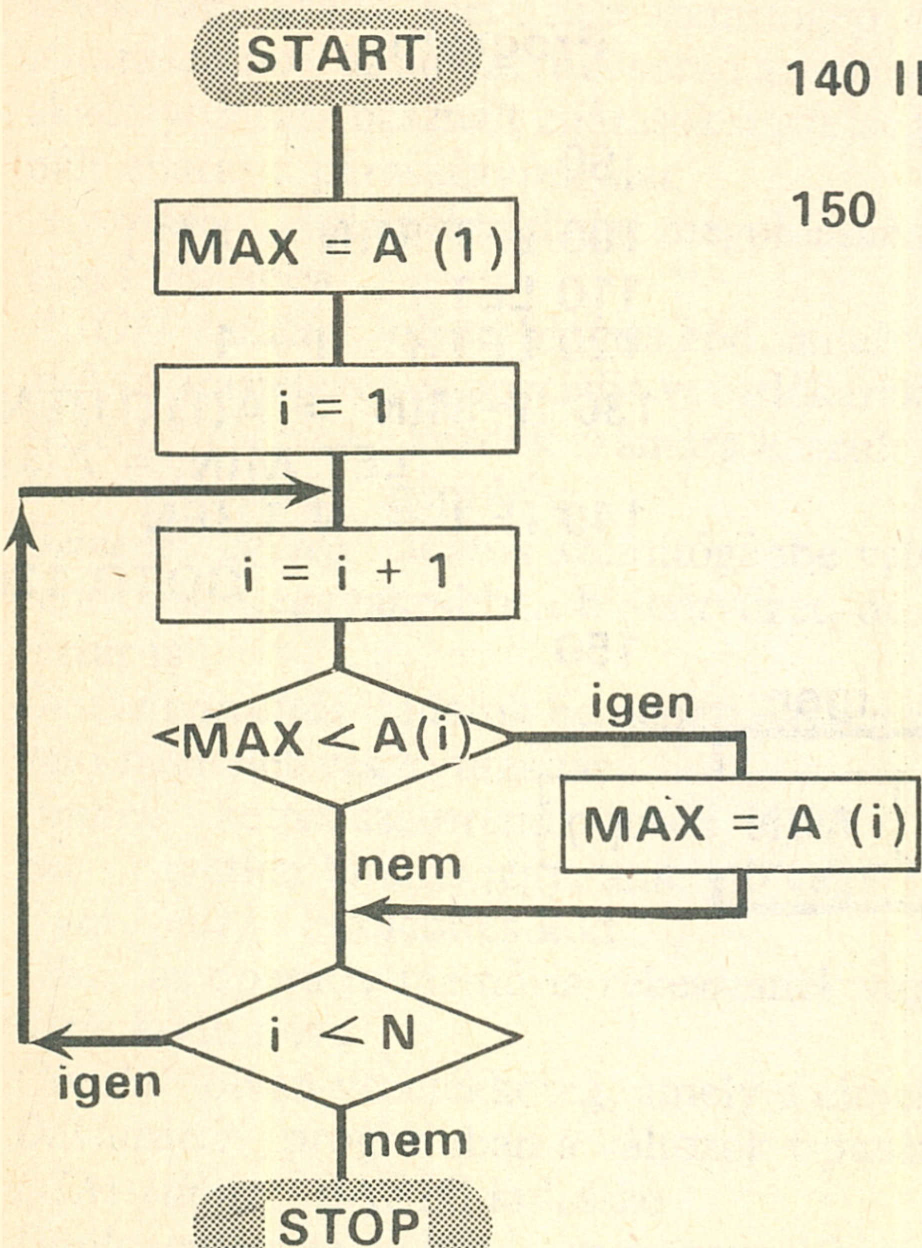
Program:

```
10 DIM A(20)
20 LET I = 0
30 LET I = I + 1
40 INPUT A (I)
50 IF I < 20 THEN GOTO 30

60 LET I = 0
70 LET AT = 0
80 LET I = I + 1
90 AT = AT + A (I)
100 IF I < N THEN GOTO 80
110 LET AT = AT/20
120 PRINT „ATLAG=''; AT
130 END
```

2. Írj programrészletet, amely egy N elemű A vektor legnagyobb értékű elemét kiválasztja, és a MAX változóba helyezi el!

Folyamatábra:

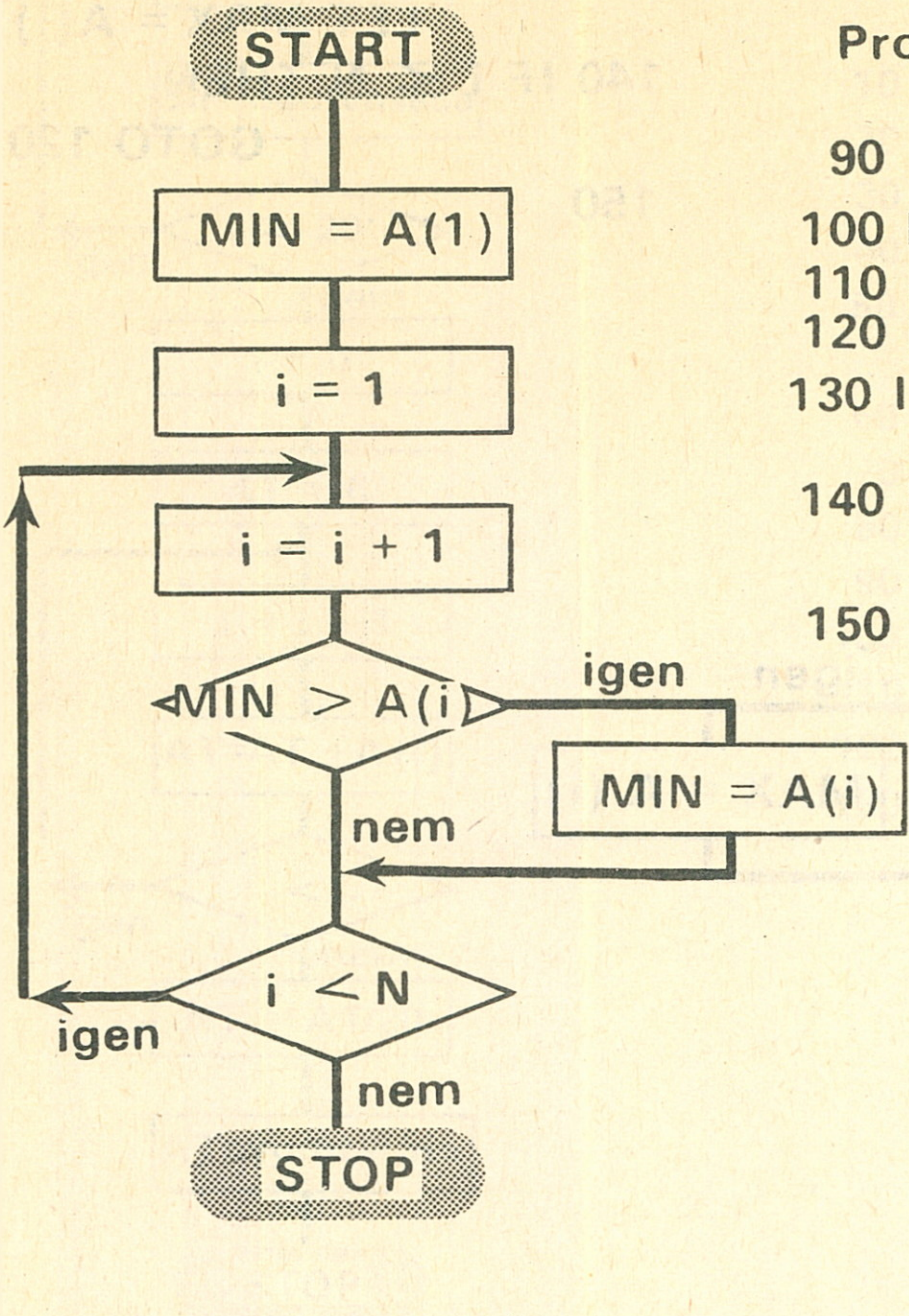


Program:

```
 90      . . .
100 LET MAX = A (1)
110 LET I = 1
120 LET I = I + 1
130 IF MAX<A (i) THEN
             LET MAX = A(I)
140 IF I < N THEN
                  GOTO 120
150      . . .
```

3. Írj programrészletet, amely egy N elemű A vektor legkisebb értékű elemét kiválasztja, és a MIN változóba helyezi el!

Folyamatábra:



Program:

```
 90       . . .
100 LET MIN = A(1)
110 LET I = 1
120 LET I = I + 1
130 IF MIN > A(i) THEN
            LET MIN = A(i)
140 IF I < N THEN
                GOTO 120
150       . . .
```

A KÖNYVBEN HASZNÁLT SZÁMÍTÁSTECHNIKAI FOGALMAK

Adat különféle dolgok vagy tények megadása számokkal vagy szavakkal

Adatsín olyan általános célú vezetékköteg, amelyen keresztül utasítások vagy adatok mennek a számítógép különböző egységei között

Adatvédelem az adatátvitel során az adatvesztés vagy torzulás megakadályozására használt eljárások összessége. A leggyakrabban használt eljárás a párosságvizsgálat

Algoritmus valamely feladat megoldásának pontos és teljes leírása, véges lépésben

ASCII (eszki) olyan bináris kód, amely kis- és nagybetűket, számokat, írásjeleket és speciális vezérlőkaraktereket tartalmaz

BASIC (bézik) kezdők számára készült programnyelv

Belső ábrázolás gépi kód

Bemeneti egység adatok számítógépbe való juttatására szolgáló berendezés. Leggyakrabban billentyűzet, de ~ ként működhet a háttértár is

Beviteli utasítás adatok bemeneti egységről az operatív memóriába való bevitelét végző utasítás

Bináris kettes számrendszerbeli

Bit egyetlen bináris jegy, amely 0 vagy 1 értéket vehet fel

Byte (bájt) nyolcbites kód

Cím az operatív memória rekeszeinek vagy egyes egységeknek bináris kódú neve

Címsín olyan vezetékköteg, amely a címek átvitelére szolgál

Deklaráció programban a változók típusának megadása

DIM (dim) BASIC kulcsszó

Duplapontosságú változó maximum 12 számjegyű számot tartalmazó változó a BASIC-ben

Egyszeres pontosságú változó maximum 6 számjegyű számot tartalmazó változó a BASIC-ben

END (end) BASIC kulcsszó

Értékadó utasítás olyan utasítás, amelynek hatására egy tárrekesz (változó) tartalma változtatás nélkül áttöltődik egy másikba

ÉS kapu olyan logikai áramkör, amely a bemenetek között ÉS kapcsolatot valósít meg

Feladatmegoldás sorrendje a feladatmegoldás egyes lépéseinek logikus menete

Feldolgozóegység a számítógép azon része, amely az adatokkal a műveletet elvégzi

Feltétel nélküli vezérlésátadó utasítás a program végrehajtása az utasításban megadott sorszámú utasítással folytatódik

Feltételes vezérlésátadó utasítás az utasításban lévő feltétel teljesülése esetén az utasítás végrehajtódik, ellenkező esetben a soron következő utasítás lép érvénybe

Folyamatábra az egyes tevékenységek időbeli sorrendjét és kapcsolatát áttekinthető formában bemutató ábra

Fordítóprogram valamely forrásnyelven leírt programból gépi kódú programot készítő program

Forrásnyelv valamely program ember által történő megírásához készült programnyelv

Forrásnyelvi utasítás forrásnyelvi szabály a programozáshoz

Gépi kód a számítógép, illetve a mikroprocesszor által közvetlenül értelmezhető jelrendszer, bináris kód

Gépi kódú program gépi kódban megírt program

GOTO (gotu) BASIC kulcsszó, jelentése „menj" vagy „lépj a..."

Háttértár mágneslemezes vagy mágnesszalagos kivitelű, nagy tömegű adat tárolására szolgáló berendezés

IF (if) BASIC kulcsszó, jelentése „ha"

INPUT (input) jelentései: 1. bemenet,
2. bevitel (BASIC kulcsszó)

K (ká) 1024 byte-ot jelölő egység

Karakter számjegy, betű és egyéb írásjel gyűjtőneve

Kettes számrendszer a 2-es alapszámon felépülő helyiértékes számrendszer

Kifejezés változók közötti műveleteket leíró képlet

Kimeneti egység adatok számítógépből való kivitelére szolgáló berendezés

Kiviteli utasítás az adat kimeneti egységen történő megjelenítését végző utasítás

Komplemens kiegészítő

Kód megállapodás szerinti jelek rendszere

Központi egység a számítógép legfőbb része, amely a vezérlő- és feldolgozóegységet és az operatív tárat foglalja magában

Kulcsszó programnyelvi utasítás, „kódolás”, azonosító része

LET (let) BASIC kulcsszó, jelentése „legyen”

Logikai áramkör logikai művelet elvégzését megvalósító áramkör

Logikai hálózat logikai áramkörökből felépített összetett áramkör

Matematikai logika a matematika egyik részterülete, amely a logikus gondolkodás formális tulajdonságait matematikai módszerekkel tárgyalja

Mbyte (megabájt) 1024 kbyte

Mikroszámítógép a legegyszerűbb felépítésű, többnyire egyetlen mikroprocesszort tartalmazó számítógép, amely többnyire pusztán gépi kódban programozható

NEW LINE (nyu lájn) utasítás végét jelző jelet előállító billentyű egyes gépeken

Nyelv elemei a nyelvben használt fogalmak (kulcsszó, utasítás, változó, kifejezés stb.)

Operatív memória a működő program számára szolgáló tárolóberendezés a számítógépben

PRINT (print) BASIC kulcsszó, jelentése: „nyomtass”

Programozás a program elkészítése

RAM (ram) írható, olvasható memória

READY (redi) a fordítóprogram készenléti állapotát jelentő felirat – magyarul: „kész”

Regiszter általános célú, általában egy gépi szó befogadására alkalmas tároló

RETURN (ritörn) utasítás végét jelző jelet előállító billentyű egyes gépeken

ROM (rom) csak olvasható, megváltoztathatatlan tartalmú memória

Software (szoftver) egy számítógéphez tartozó vagy illeszthető (vásárolható) programok gyűjtőneve

Szabályhalmaz olyan halmaz, amelynek szabályok az elemei

Számítógép olyan elektronikus berendezés, amely adatokon végez műveleteket, működéséhez program szükséges, és saját működését is vezérelni tudja

Személyi számítógép kis méretű, olcsó, általános célú számítógép, amely magas szintű nyelven, illetve nyelveken programozható (BASIC, Fortran, C, PASCAL stb.)
Szóhosszúság a memóriarekesz hossza bitekben vagy byte-okban
Szöveg karaktersorozat
Szöveges változó karaktersorozat tárolására szolgáló memóriarekesz vagy memóriarekeszek
Tárolóelem 1 bit tárolására képes áramkör
THEN (den) BASIC kulcsszó
Tizenhatos számrendszer a 16 alapszámon felépülő, a byte-szervezésű számítógépeknél használatos helyiértékes számrendszer
Tízes számrendszer a 10 alapszámon felépülő, a mindennapi életben használatos helyiértékes számrendszer
VAGY kapu olyan logikai áramkör, amely a bemenetek között VAGY kapcsolatot valósít meg
Változó adat tárolására kijelölt memóriarekesz vagy memóriarekeszek
Változónév a programozó által megadott név, amely a programban a változó azonosítására szolgál
Változótípus egy adott tulajdonságú változó
Vektor névvel ellátott számsorozat
Vezérlőegység a központi egység azon része, amely a számítógép egyes részeinek munkáját összehangolja, és biztosítja az automatikus működést
Vezérlőjel-vezeték a vezérlőegység jeleit hordozó vezeték
Végjel egy utasítás végét jelző jel a BASIC nyelvben
\# *(hes márk)* BASIC jelölés egy változótípushoz

TARTALOM

SZIKSZAI CSABA

Barátunk, a számítógép

A könyv első vállalkozás a Móra Kiadó történetében. Azoknak a fiataloknak nyújt segítséget, akik valamelyik iskolai szakkörben most ismerkednek a számítástechnika alapjaival, vagy készülnek erre az ismerkedésre. A szerző gyerekekkel beszélget, és közben fokról fokra kiderül, hogyan is működik az a kis masina, melynek gombjait nyomogatva már bizonyára sokszor szórakoztak különféle videojátékokkal, megjelenítve azokat tévéjük képernyőjén.
Az ismerkedéshez semmiféle előzetes tudásra nincs szükség – csak egy kis figyelemre. S ennek jutalma, hogy a kis olvasók a végén már különféle feladatokat tudnak kitalálni és megfogalmazni a gép számára.

MÓRA FERENC KÖNYVKIADÓ

HU ISSN 0236-980 X

ISBN 963 11 3899 2

Móra Ferenc Ifjúsági Könyvkiadó, Budapest
Felelős kiadó: Sziládi János igazgató
Zrínyi Nyomda (85.1012/10), Budapest, 1985
Felelős vezető: Vágó Sándorné vezérigazgató
Felelős szerkesztő: Simonffy Géza
Szakmailag ellenőrizte: Németh Pál
Műszaki vezető: Szakálos Mihály
Képszerkesztő: Gáspár Imre
Műszaki szerkesztő: Szántai Ágnes
63 000 példány
Terjedelem: 12,16 (A/5) ív + 8 oldal melléklet
IF 5430

32,–Ft

hobbi

MÓRA



